VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

INTERAKTIVNÍ WEBOVÉ AUDIO APLIKACE PRO PODPORU VÝUKY
INTERACTIVE WEB APPLICATIONS SUPPORTING EDUCATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Filip Kratoš**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Mgr. Pavel Rajmic, Ph.D.**
SUPERVISOR

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Audio inženýrství**
Ústav telekomunikací

Student: Filip Kratoš ***ID:*** 174458
Ročník: 3 ***Akademický rok:*** 2016/17

NÁZEV TÉMATU:

Interaktivní webové audio aplikace pro podporu výuky

POKYNY PRO VYPRACOVÁNÍ:

V bakalářské práci půjde o interaktivní podání algoritmů používaných v oblasti zpracování zvuku, konkrétně: Kombinace (míchání) signálů, Součet kosinus a sinus (a změna fáze), Filtrace signálu, Význam fáze u audiosignálu, Aliasing a jeho projevy, Rozdíl mezi lineárními a nelineárními systémy. Nastudujte problematiku vztahující se k výukovým apletům, jejichž funkcionalitu a ovládání navrhnete, implementujete a otestujete. Ke každému apletu vytvořte webovou stránku s výběrem příslušné teorie.

DOPORUČENÁ LITERATURA:

[1] Oppenheim, A.: Signals & Systems. PHI, 2. vydání, 208. ISBN 978-8120312463.

[2] Smékal, Z.: Analýza signálů a soustav. Skriptum. VUT v Brně, 2012.

Termín zadání: 1.2.2017 ***Termín odevzdání:*** 8.6.2017

Vedoucí práce: doc. Mgr. Pavel Rajmic, Ph.D.
Konzultant:

doc. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

Bakalářská práce je zaměřena na návrh webových apletů v jazyce javascript. Zabývá se návrhem a vytvořením šesti apletů z oblasti jednorozměrných signálů, především audiosignálů. Konkrétně to jsou aplety na téma kombinace signálů, součet funkce sinus a kosinus, filtrace signálu, vliv fáze na audiosignál, aliasing a jeho projevy a lineární a nelineární systémy. Práce obsahuje shrnutí teoretických podkladů pro tvorbu apletů, popis implementace a popis jejich rozhraní.

KLÍČOVÁ SLOVA

Aplet, Javascript, Signál, Audiosignál, Harmonická funkce, Filtrace, Aliasing

ABSTRACT

This bachelor's thesis is focused on design of web applets using Javascript. It deals with design and creation of six applets works with one-dimensional signals, especially audio-signals. The applets are Combination of Signals, Summation of Sine and Cosine Functions, Signal Filtering, Effect of Phase on Audio-signal, Effect of Aliasing, and Linear and Nonlinear Systems. The thesis includes theoretical background for applets, describing of implementation and describing of user interface of applets.

KEYWORDS

Applet, Javascript, Signal, Audio-signal, Harmonic function, Filtering, Aliasing

KRATOŠ, F. *Interaktivní webové audio aplikace pro podporu výuky*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2017. 40 s. Vedoucí bakalářské práce doc. Mgr. Pavel Rajmic, Ph.D..

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma Interaktivní webové audio aplikace pro podporu výuky jsem vypracoval samostatně pod vedením vedoucího semestrální práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené semestrální práce dále prohlašuji, že v souvislosti s vytvořením této semestrální práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne ....

(podpis autora)

PODĚKOVÁNÍ

Děkuji vedoucímu bakalářské práce doc. Mgr. Pavlu Rajmicovi, Ph.D., za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

V Brně dne ....

(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

ÚVOD

V dnešním světě plném moderních technologií je potřeba i do škol zavádět interaktivní prvky výuky. Nemusí to ani být přednášky přenášené přes internet a podobné revoluční metody. Pro pochopení mnoha témat je velice výhodné vyzkoušet si je sám i prakticky. Jednou z možností jsou laboratorní cvičení, které jsou ale náročné na přípravu a čas. Velice praktické se ukazují být počítačové aplety, které s několika jednoduchými ovládacími prvky dokážou nasimulovat nějaký jev a názorně jej předvést.

Cílem této práce je návrh a vytvoření šesti webových apletů v jazyce Javascript. Aplety jsou zaměřeny na téma analýza signálů, zvláště pak na zvukové signály. Jejich cíl je interaktivní podpora výuky v předmětu Analýza signálů a soustav (BASS). Pět z těchto šesti apletů již před napsáním práce v nějaké formě existovalo jako výstup závěrečných prací jiných studentů Ústavu telekomunikací. U šestého apletu se jedná o zcela nový návrh.

Původní aplety jsou vytvořeny v jazyce Java, který je dnes pro provozování webových apletů nevhodný, vzhledem k bezpečnostní politice webových prohlížečů, které mají tendenci Javu tvrdě blokovat. V aktuálních verzích většiny webových prohlížečů dokonce již žádným způsobem není možné takovéto aplety spustit.

První kapitola popisuje teoretické základy jazyka Javascript a hlavní použitou knihovnu funkcí JSXGraph. Obsahuje také srovnání Javascriptu s jinými programovacími jazyky a odůvodnění, proč byl vybrán.

Ve druhé kapitole je popsán aplet Kombinace signálů. Funkčně vychází z původního apletu ing. Riška a odstraňuje některé nedostatky jeho uživatelského prostředí. Ukazuje lineární kombinaci tří základních signálů, popřípadě bílého šumu. Signálům je možné zadat do součtu různou váhu, tu je také možné nastavit konvexně v trojúhelníkových souřadnicích.

Třetí kapitola se zabývá apletem ukazujícím součet dvou harmonických funkcí o stejné frekvenci. Původní aplet doktora Zdenka Průši je přehledný a funkční. Obsahuje pouze chybu ve výpočtu koeficientu, která zde byla opravena. Je zvolen také jiný způsob zadávání hodnot, pomocí posuvníků. Aplet samotný obsahuje dva grafy. Jeden vstupní s dvěma signály a jeden výstupní. Ukázáno je, že součet harmonických funkcí o stejné frekvenci dává opět harmonickou funkci o stejné frekvenci.

Čtvrtá kapitola popisuje aplet se zabývající se filtrací audiosignálu. Původní aplet ing. Luďka Novotného je nejpovedenější ze všech již existujících apletů. Bylo zde tedy cílem vytvořit co nejpřesnější repliku. Aplet obsahuje vstupní sekci s možností

volby signál, filtr s nastavitelnými parametry a výstupní sekci. Vstupní a výstupní signál je možné porovnat na grafech průběhu a spekter, případě poslechem.

Tématem pátek kapitoly je význam fáze u audio signálu. Vytvořený aplet vychází ze staršího apletu doktora Novotného Ukazuje rozklad na jednotlivé harmonické složky a opětovnou rekonstrukci obdélníkového signálu s volitelnou střídou.

V šesté kapitole je popsán aplet zaměřený na aliasing a jeho projevy. Původní aplet ing. Průši byl opět vcelku povedený. Tento aplet se mu tedy snažil přiblížit. Ukazuje graf spektrálních čar navzorkovaného sinusového signálu a demonstruje vzniklou periodizaci spektra, která v případě nedodržení minimální vzorkovací frekvence může vést ke zkreslení signálu.

Sedmá kapitola se zabývá návrhem zcela nového apletu ukazujícího rozdíl mezi lineárním a nelineárním systémem a jeho aplikaci na dva signály před a po jejich součtu.

Cílem této práce je nastudování teoretických podkladů pro tyto aplety a jejich následná implementace a otestování.

1 JAVASCRIPT

1.1 Historie

Javascript je skriptovací programovací jazyk, jehož primární funkce je interaktivita obsahu webových stránek. Původní jazyk Javascript byl vyvinut v roce 1996 programátorem Brendanem Eichem ze společnosti NetScape. Ta mimo jiné vyvíjela internetový prohlížeč NetScape, který v té době měl majoritní podíl na trhu, hned za Internet Explorerem od Microsoftu.

Právě Microsoft o rok později přišel se svou vlastní verzí nazvanou JScript, která doplňovala původní Javascript a odstraňovala některé jeho chyby. Následná spolupráce obou firem vyústila ve standardizaci jazyka pod jménem ECMA-262, vydanou Evropskou asociací výrobců počítačů (EMCA). Aktuální verze z června 2016 nese označení 6.0.

Problémem této standardizace je její přílišná volnost, která hlavně v prvních letech po uvedení vedla k rozdílné interpretaci některých částí kódu v různých prohlížečích. Dnes již tyto rozdíly nejsou tak významné, avšak zcela tento problém nevymizel.[1]

1.2 Vlastnosti jazyka

Javascript je nekompilovaný skriptovací jazyk pro webové stránky, prováděný na straně klienta. V praxi to znamená, že ze serveru se odešle pouze napsaný skript, a jeho interpretace a vykonání všech požadavků je úkolem webového prohlížeče, který uživatel používá k zobrazení stránky. Výhodou je, že není potřeba mít podporu jazyka implementovanou přímo na serveru, což zjednodušuje tvorbu webu, hlavní nevýhodou pak je určitá neovlivnitelnost konečného výsledku ze strany autora webu, která je závislá na uživatelově zařízení a jeho softwaru.

Javascript je komplementární jazyk, což znamená, že sám o sobě nemá žádný význam. Smyslu nabývá pouze ve spojení s jazykem HTML, do jehož syntaxe je začleněn, stejně jako třeba jazyk CSS starající se o vizuální část stránek [2].

Je to také jazyk do značné míry zjednodušený, stojící až nad kompilovanými jazyky, ze kterých vychází jeho syntaxe, ale vynechává konstrukce, které nejsou nezbytné. Příkladem může být deklarace proměnné, která nevyžaduje určit typ proměnné, prohlížeč jej sám později definuje podle hodnoty do něj uložené. To vede

sice ke zjednodušení programování, ale znemožňuje to použití javascriptu na rozsáhlé a komplikované projekty.

```
var A = 0      //deklarace a inicializace proměnné v javascriptu
int A = 0;     //deklarace a inicializace proměnní v Javě
```

Je to jazyk objektový, ale ne s takovými možnostmi, jak je obvyklé. Nemá podporu tříd a jejich dědičnosti. Hlavní objektové vlastnosti jazyka spočívají v tzv. Objektovém modelu dokumentu (DOM). To je rozhraní, kterým javascript komunikuje s prohlížečem. Umožňuje mu přístup k oknům a manipulaci s nimi a k vlastnostem obrazovky (např. zjištění rozlišení). [2].

1.3 Interakce s HTML

Existují tři základní způsoby zápisu javascriptu do HTML dokumentu. Základní způsob je zapsat skript v jednom kuse na začátku, v hlavičce dokumentu. Skript se uvozuje HTML tagem `<script>` a ukončuje jeho párovým znakem `</script>`. Takto se většinou zapisují deklarace funkcí a skripty, jež se mají provést ihned při načtení stránky.

V některých případech je možné umístit blok kódu javascriptu i doprostřed HTML kódu v těle dokumentu. Vzhledem k tomu, že prohlížeč webovou stránku načítá postupně, přesně tak, jak je zapsaná, příkaz se vykoná v místě, kde je zapsaný.

Další možnost je obdoba té první, pouze se skript umístí do externího souboru s koncovkou *.js, a do hlavičky dokumentu se uvede odkaz na něj. To jednak výrazně zpřehledňuje kód a jednak je to praktické při vytváření více jednotlivých stránek, které využívají stejné skripty.

Poslední možností je umístění skriptu přímo jako atribut HTML elementu. To se využívá často u skriptů, které jsou přímo na tento element navázány. U tohoto způsobu použití jsou nezbytnou součástí tzv. události. Ty představují mechanismus javascriptu, kterým reaguje na činnost uživatele. Umožňují tak spuštění skriptu i po načtení stránky. Běžnou událostí je například kliknutí či přejetí myší přes element.

Typickým příkladem skriptu vázaného na HTML element může být třeba změna obrázku po najetí myší na něj. Kód pak bude vypadat takto:

```
<img src="obrazek1.jpg" onmouseover="this.src='obrazek2.jpg'">
```

Modře označená část kódu představuje skript. `Onmouseover` je událost přejetí myší přes element, `this` je odkaz na objekt, který skript zavolal, tedy HTML element obrázku a `src` představuje zdroj, ze kterého se má nový obrázek načíst.

1.4 Proč javascript

Léta se na Ústavu telekomunikací i téměř kdekoliv jinde na tvorbu webových apletů využíval jazyk Java. Jeho hlavní výhoda spočívala v jednoduchosti implementace a množství volně dostupných knihoven schopných zajistit téměř jakoukoliv operaci s minimálním úsilím programátora. Navíc je Java přenositelná mezi jakýmikoliv platformami.

To vychází z principu jejího fungování, kdy program tak, jak je vytvořený a distribuovaný, není úplně zkompilovaný. Je jen předpřipravený na kompilaci, o kterou se při spuštění programu stará tzv. Java Virtual Machine, což je určitý spojovací článek mezi programem a hardwarem, na kterém program běží. Ten se postará o překlad programu přímo určený pro platformu, na které běží.

Díky tomu je Java velice rozšířená v různých mobilních zařízeních a terminálech. Přináší to však i značná bezpečnostní rizika. Program v Javě, který je spouštěný z internetu je velice těžké zabezpečit, je to tudíž snadná cesta do počítače pro škodlivý software. V dnešním světě, kdy jsou na bezpečnost internetu kladeny vysoké nároky, přistupují webové prohlížeče k Java apletům velice podezřívavě a snaží se je všemožně blokovat. Díky tomu i v případě, že si jsme původem apletu a jeho bezpečností jistí, je velice obtížné docílit jeho spuštění. Kromě toho snížení úrovně zabezpečení, které je pro spuštění nezbytné, zase může být nebezpečné při jiných situacích.

Proto se nabízí jako řešení tvořit aplety v jazyce, který je pro webové prohlížeče přirozený a jeho bezpečnost přímo vychází z jeho specifikace a implementace v prohlížečích. Takových jazyků je dnes několik. Jak již bylo naznačeno v kapitole 1.2, rozdělují se na dvě základní skupiny.

Za prvé jsou to serverové jazyky. Výpočet všech operací probíhá na straně serveru a do počítače uživatele se posílají pouze výsledná data potřebná pro vykreslení stránky. To sice umožňuje vytvoření složitějších programů, jejichž činnost není závislá na výkonu počítače uživatele, na druhou stranu tento způsob klade vyšší nároky na rychlost připojení k internetu a množství přenesených dat. Mimo to je třeba provozovat web na serveru, který tento jazyk podporuje. Navíc je v takovém případě nutné účinně ošetřit bezpečnost serveru. V praxi se tedy tento přístup využívá hlavně v kombinaci

s rozsáhlými databázemi, ke kterým je potřeba při vykreslování stránek přistupovat. Příkladem takového jazyka je PHP. [2]

Na tvorbu webového apletu je tedy vhodnější klientský jazyk, který přes web přenáší pouze zdrojový kód a všechny operace jsou zpracovány počítačem uživatele. Pro vývoj apletů v rámci této práce byl zvolen Javascript, který představuje dnes nejrozšířenější a nejvíce podporovaný klientský jazyk.

1.5 Knihovny v javascriptu

Jakýkoliv programovací jazyk, ať je složitý a rozsáhlý nebo obsahuje jen nejzákladnější prvky, je vždy pouze určitým souborem mechanismů a prostředků pro základní práci s proměnnými, operátory a objekty. S jejich pomocí se dá dosáhnout jakéhokoliv výsledku, který je v daném jazyce možný, ovšem naprogramovat všechny funkce pro každý projekt znovu by bylo zbytečně složité a nesmyslné. Proto pro každý jazyk existují tzv. knihovny, které shrnují množství již hotových funkcí, většinou podobného zaměření. Programátor je pak pouze vhodně zasadí do svého kódu a většinou se nemusí starat o jejich vnitřní strukturu.

Některé knihovny bývají přímo součástí vývojových prostředí. Ty většinou zajišťují běžné matematické operace nebo se starají o grafické uživatelské prostředí. Jiné je třeba do projektu vložit jako externí soubory. Mohou být k dispozici zdarma, u jiných je jejich použití zpoplatněno.

Tak jako i pro každý jiný jazyk, i pro Javascript existuje množství knihoven. Nejběžněji používanou externí knihovnou je jQuery. Ten ale není pouze souborem mnoha funkcí, nýbrž tzv. framework. Framework by se dal zjednodušeně popsat jako samostatný programovací jazyk vytvořený v rámci jiného programovacího jazyku, který značně rozšiřuje jeho možnosti. [3]

1.5.1 Knihovna JSXGraph

Pro tuto práci bylo potřeba v prvé řadě vybrat knihovnu pro vykreslování grafů. Existuje jich vcelku velké množství. Vybrána byla knihovna JSXGraph. Je dostupná pod licencí LGPL, která umožňuje bezplatné použití celé knihovny nebo libovolné její části pro jakýkoliv komerční i nekomerční projekt.

Tato knihovna v sobě integruje nástroje pro vykreslování grafů libovolných funkcí a diskrétních grafů, vykreslování všemožných geometrických objektů a rozhraní pro práci s nimi. Navíc je schopna umožnit uživateli pohyb s těmito objekty pomocí myši nebo dokonce tímto způsobem získat vstupní data (např. přečtením aktuálních

souřadnic bodu). Vše se samozřejmě vykresluje v reálném čase. Navíc je práce s ní velice intuitivní.

Na webových stránkách projektu [4] je k dispozici množství ukázkových příkladů s vysvětlením a zdrojovým kódem pro snadnější pochopení. Na stránkách [5] je pak k dispozici kompletní specifikace všech funkcí v JSXGraph a všech jejich parametrů.

Základním prvkem knihovny jsou rámce nazývané *board*. Každý rámec funguje jako samostatný oddíl, je však možné je k sobě řetězit pomocí funkce *addChild*, například:

```
board1.AddChild(board2),
```

rámec `board2`, který je jako parametr funkce pak zdědí přístup k funkcím a objektům rámce `board1`.

Každý rámec má svůj vlastní souřadnicový systém, pomocí kterého jsou do něj umísťovány objekty.

2 APLET KOMBINACE SIGNÁLŮ

2.1 Lineární kombinace

Jednou ze základních operací s jakýmikoliv signály je jejich sčítání. Představme si dva vstupní signály $f(t_1)$ a $f(t_2)$, například dva audio signály ze dvou různých mikrofonů. Pokud je sečteme a přivedeme na jeden výstup, bude mít výsledný signál v každém okamžiku hodnotu součtu hodnot obou signálů. Každý signál pak může být násoben koeficientem u, který představuje váhu tohoto signálu.

Rovnice popisující tento jev bude mít tvar

$$f(x) = u_1 f(x_1) + u_2 f(x_2), \tag{2-1}$$

obecně pro *n* signálů pak

$$f(x) = \sum_{i=1}^{n} u_i f(x_i). \tag{2-2}$$

Operace sčítání signálů je základem činnosti zařízení zvaného mixážní konzole nebo také zkráceně „mixpult“, které se používá při ozvučování nebo nahrávání audio signálu. Do jednotlivých vstupů mixážní konzole je možné zapojit různé zdroje signálu, těm je pak nastavena váha pomocí tahového potenciometru, kterému se říká fader [čti fejdr]. Nakonec jsou všechny signály sečteny pomocí operačního zesilovače v invertujícím zapojení a přivedeny na hlavní výstup. [6][7]

Obecně lze lineární kombinaci pojmout jako součet vektorů v *n*-rozměrném prostoru. Signály, které jsou jednorozměrné pak lze chápat jako vektory ležící na ose *y*, které v čase mění svou velikost. Velikost vektoru výsledného signálu lze získat prostě, jako součet délek jednotlivých vektorů.

Na ose *x* bývá obvykle znázorněn čas. Jakýkoliv signál tedy můžeme vyjádřit v grafu jako závislost výchylky na čase.

2.2 Konvexní kombinace

Konvexní kombinace je speciálním případem lineární kombinace. Platí u ní, že váhy všech signálu dohromady dávají hodnotu 1 a zároveň jsou všechny jejich koeficienty nezáporné. [8]

$$u_1 + u_2 + \ldots + u_n = 1 \qquad \text{pro } u_i \geq 0 \tag{2-3}$$

2.3 Popis apletu

Aplet se skládá ze tří vstupních sekcí představujících jednotlivé kanály (viz Obrázek 2.1). V každé z nich je možné vybrat jeden z těchto čtyř předdefinovaných signálů: Sinus s frekvencí $1/2\pi$, lichý obdélníkový pulzní signál se střídou i frekvencí 0,5, pilový signál s frekvencí 1 a bílý šum, vygenerovaný vždy při spuštění apletu zvlášť pro každý kanál.

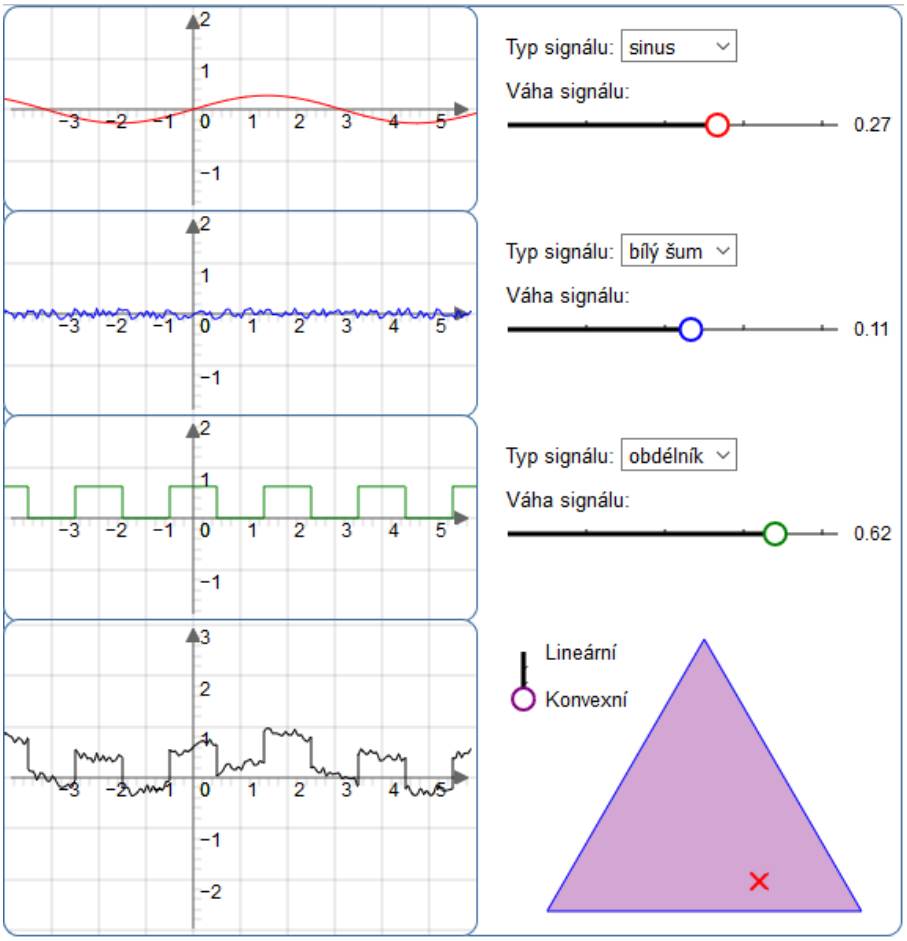
Šum se z důvodu přehlednosti grafů generuje pouze pro sto padesát bodů, tedy prakticky pro každý druhý bod z horizontálního rozlišení grafů, které činí 300px. Skutečný šum sice vždy má zcela náhodnou hodnotu pro každý bod, takto vykreslený graf by ale vypadal jako nepřehledná změť čar a názornosti apletu by to uškodilo.

Kromě pole pro výběr typu signál pak každá sekce obsahuje posuvník pro nastavení váhy signálu v rozsahu -1 až 1 a graf, ve kterém se zvolený signál zobrazí. Sekce jsou pro přehlednost umístěny pod sebou a jsou od sebe barevně odlišeny barvou vykresleného signálu a táhla posuvníku.

Pod vstupními grafy se nachází čtvrtý graf, který zobrazuje výsledný signál vzniklý lineárním součtem všech tří vstupních signálů. Je vykreslen černou barvou. Oproti vstupním grafům má větší rozsah na ose y (-3 až 3 vs. -2 až 2), má ale stejné absolutní měřítko (signály se stejnou velikostí budou na obrazovce vykresleny stejně velké).

Poslední částí je sekce pro konvexní zadávání vah signálů. Ta obsahuje přepínač s polohami „lineární“ a „konvexní“. Pokud je v poloze „lineární“, tato sekce je deaktivovaná a váhy signálů jsou zadávány polohou posuvníků. V poloze „konvexní“ jsou váhy nastaveny podle umístění pohyblivého bodu ve vyznačeném prostoru trojúhelníku, kde každý roh představuje váhu 1 jednoho ze signálů a zbylé dva pak mají váhu 0.

Že je tato sekce aktivována, je signalizováno kromě polohy přepínače ještě zvýrazněním barvy prostru uvnitř trojúhelníku. Přepínat mezi způsoby zadávání je možné také tahem za ovládací prvek vztahující se ke konkrétnímu způsobu zadávání, tedy v případě, že je aktivní konvexní zadávání vah, tak tah za posuvník jej přepne na lineární a naopak, když je aktivní lineární, tak posunutí bodu jej přepne na konvexní.



Obrázek 2.1: Rozhraní apletu Kombinace signálů

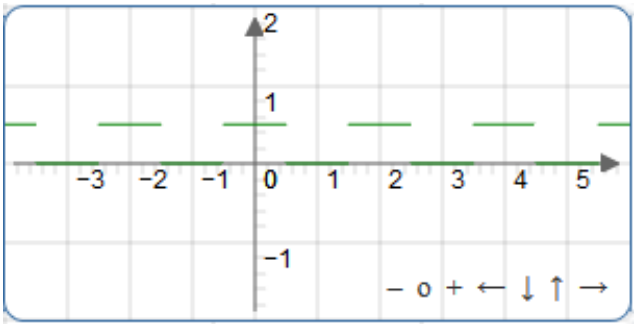
2.4 Implementace apletu

V souladu s principy knihovny JSXGraph je kód, rozdělen do několika částí, které odpovídají jednotlivým rámečkům grafického rozhraní. Tyto sekce kódu v javascriptu jsou vloženy do HTML kódu, který definuje význam jednotlivých prvků z pohledu na aplet jako na internetovou stránku.

Základem kódu je hlavní rámec, ve kterém je definována většina funkcí a v němž jsou umístěny všechny grafické prvky apletu, které jsou vytvořené javascriptem, popřípadě listenery, které reagují na události HTML prvků (mezi ně patří např. pole pro výběr signálu apod.).

Rámečky grafů dědí všechny vlastnosti hlavního rámce, čímž získávají přístup k jeho funkcím a prvkům. Samy pak obsahují vždy pouze funkci pro zobrazení grafu. Ta funguje tak, že zavolá funkci, které předá bod x, a zpátky dostane bod y. Jejím problémem ale je, že vykresluje grafy po jednotlivých bodech a nikoliv jako spojnice dvou sousedních bodů. Díky tomu v místě nespojitostí např. u obdélníkového signálu

čáry grafu nejsou spojeny (Obrázek 2.2). Matematicky je to sice správně, ovšem pro přehledné zobrazení signálu není tento způsob vhodný. Bylo proto nutné v případě, že konkrétní signál obsahuje nespojitosti, svislé čáry dokreslit do grafu zvlášť.



Obrázek 2.2: Nespojitě vykreslený graf

Větším problémem bylo vytvoření ovládání, pro konvexní způsob zadávání signálu. Knihovna JSXGraph obsahuje jednoduché funkce pro vytvoření jakéhokoliv polygonu i bodu. Stejně tak objekt bodu má vestavěné metody pro zjištění jeho polohy v rámci, např. pro zjištění polohy na ose x bodu `point1` se použije metoda:

```
x = point1.X(),
```

ovšem není implementován žádný způsob omezení rozsahu pohybu bodu v polygonu. Bylo tedy nutné vytvořit funkci, která zabrání bodu dostat se mimo trojúhelník.

To je řešené funkcí, která pro každou z přímek vymezujících plochu trojúhelníku zjistí, zda je bod nad ní nebo pod ní. Při zahrnutí všech kombinací může nastat celkem sedm případů. Buď je bod v trojúhelníku a pak se nestane nic nebo se nachází v jedné z výsečí nad vrcholem, pak je přesunut do tohoto vrcholu, nebo je jinde mimo trojúhelník a pak je bod projektován na nejbližší bod nejbližší úsečky.

Ze souřadnic bodů $A[A_x;A_y]$ a $B[B_x;B_y]$ definujících přímku je odvozen směrový vektor $\vec{\mathbf{p}} = (B_x - A_x; B_y - A_y)$ a normálový vektor $\vec{\mathbf{n}} = (B_y - A_y; A_x - B_x)$. Z těch je na základě rovnice

$$ax + by + d = (B_y - A_y)x + (A_x - B_x)y + (b \cdot p_x - a \cdot p_y) = 0, \tag{2-4}$$

kde p_y a p_y jsou souřadnice posunovatelného bodu [9], odvozena rovnice přímky ve tvaru,

$$y = \frac{A_x(B_y - A_y) + A_y(A_x - B_x) + x(A_y - B_y)}{A_x - B_x}. \tag{2-5}$$

Stejným postupem je odvozena i rovnice kolmice k této přímce procházející aktuální pozicí posunovatelného bodu. Pouze je prohozen normálový vektor se směrovým vektorem. Následně je pomocí soustavy rovnic odvozen průsečík těchto přímek. Do něj bude poté přesunut bod. Pro souřadnici *x* tohoto průsečíku pak dostaneme rovnici:

$$x = \frac{p_x(A_x - B_x)(B_x - A_x) + p_y(B_y - A_y)(A_x - B_x) + A_x(B_y - A_y)^2 + A_y(B_y - A_y)(A_x - B_x)}{(B_x - A_x)(A_x - B_x) + (B_y - A_y)(A_y - B_y)}. \quad (2\text{-}6)$$

Bod *y* k němu pak získáme opět pomocí rovnice (2-5).

3 APLET SOUČET SINUS A KOSINUS

3.1 Součet funkcí sinus a kosinus

Funkce sinus, potažmo kosinus jsou základní funkce, na které lze rozložit jakýkoliv signál. Označují se také jako harmonické funkce. Periodické funkce se rozkládají na harmonické funkce – složky pomocí Fourierových řad, kdy všechny harmonické složky mají frekvenci, jež je násobkem frekvence základní složky. U neperiodických funkcí se používá Fourierova transformace a výsledkem jsou harmonické složky o různých frekvencích.

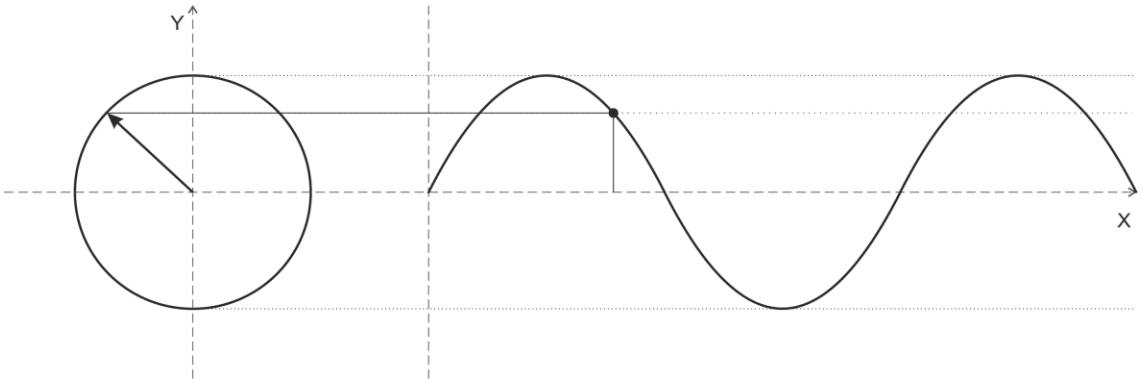
Opačně, pokud se skládají harmonické složky o frekvencích, které nemají společného dělitele, vzniká neperiodický pohyb a při sčítání harmonických složek, které jsou navzájem násobky, vzniká periodický pohyb. Sečtou-li se ale harmonické funkce o stejné frekvenci, výsledná funkce bude rovněž vždy harmonická. Bude mít pouze jinou velikost a fázi [10].

3.2 Průvodiče harmonických funkcí

Mějme úsečku. Pokud ji necháme rotovat kolem jednoho jejího krajního bodu a budeme sledovat pohyb druhého krajního bodu v jedné z os, získáme harmonickou funkci (Obrázek 3.1). Takovou úsečku nazýváme polohový vektor nebo průvodič.[9]

Často se používají průvodiče umístěné do středu komplexní roviny, které znázorňují průběh střídavého napětí a proudu. Pomocí průvodiče ale lze vyjádřit jakoukoliv harmonickou funkci. [11]

Součet harmonických funkcí lze odvodit jako vektorový součet průvodičů těchto funkcí.



Obrázek 3.1: Průmět průvodiče harmonické funkce na osu y

3.3 Odvození rovnic

Jednodušší případ nastane, když mají funkce různou velikost, ale nemají posunutou fázi, tedy platí:

$$\sin(x+\varphi)=\sin(x)=0 \Leftrightarrow x=2k\pi, \tag{3-1}$$

$$\cos(x+\varphi)=\cos(x)=0 \Leftrightarrow x=2k\pi+\frac{\pi}{2}. \tag{3-2}$$

Při součtu dvou nebo více funkcí sinus je velikost výsledné funkce rovna součtu velikostí složek a fáze zůstává nezměněna, jak ukazuje rovnice

$$A\cdot\sin(x)+B\cdot\sin(x)=(A+B)\cdot\sin(x), \tag{3-3}$$

totéž platí i při součtu funkcí kosinus:

$$A\cdot\cos(x)+B\cdot\cos(x)=(A+B)\cdot\cos(x). \tag{3-4}$$

Při součtu funkce sinus s funkcí kosinus:

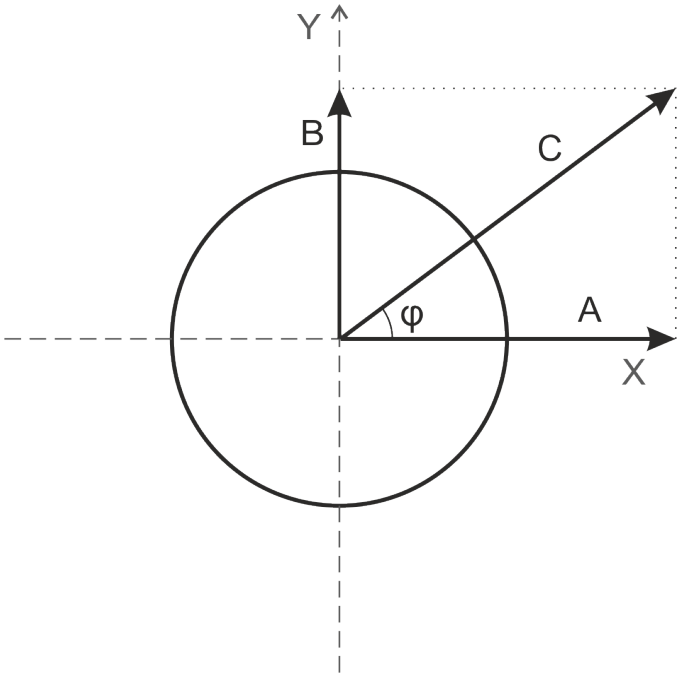
$$A\cdot\sin(x)+B\cdot\cos(x)=C\cdot\sin(x+\varphi), \tag{3-5}$$

vyjdeme z jejich průvodičů. Ty svírají pravý úhel. Můžeme si odvodit pravoúhlý trojúhelník, jehož odvěsny odpovídají průvodičům a odvěsnu tvoří průvodič výsledné funkce (Obrázek 3.2)[9]. Jeho velikost pak odvodíme podle Pythagorovy věty:

$$C=\sqrt{A^2+B^2}. \tag{3-6}$$

Z tohoto trojúhelníku odvodíme i úhel φ:

$$\tan\varphi=\frac{B}{A}. \tag{3-7}$$



Obrázek 3.2: Průvodiče harmonických funkcí sinus a kosinus

Složitější případ nastává, pokud mají i původní funkce posunutou fázi:

$$A\cdot\sin(x+\varphi_1)+B\cdot\sin(x+\varphi_2)=C\cdot\sin(x+\varphi), \tag{3-8}$$

$$A\cdot\sin(x+\varphi_1)+B\cdot\cos(x+\varphi_2)=C\cdot\sin(x+\varphi), \tag{3-9}$$

$$A\cdot\cos(x+\varphi_1)+B\cdot\cos(x+\varphi_2)=C\cdot\cos(x+\varphi). \tag{3-10}$$

Na rozdíl od předchozího případu je zde jen malý rozdíl mezi součtem dvou stejných funkcí (3-8),(3-10) a součtem sinus a kosinus (3-9), protože i ten můžeme vyjádřit jako rozdíl fází. Vyjdeme z rovnice (3-8). K odvození opět lze použít průvodiče. Ty tentokrát nebudou svírat pravý úhel. Proto se pro odvození použijí geometrické vzorce popisující rovnoběžník (Obrázek 3.3)[9].

Modul C bude odpovídat jedné z jeho úhlopříček, jejíž velikost spočítáme jako:

$$C=\sqrt{A^2+B^2+2AB\cdot\cos(\varphi_2-\varphi_1)} \tag{3-11}$$

a úhel φ pak bude odpovídat úhlu této úhlopříčky vůči ose x (součtu úhlu úhlopříčky vůči jedné ze stran a úhlu této strany vůči ose x):

$$\tan(\varphi) = \frac{A \cdot \sin(\varphi_1) + B \cdot \sin(\varphi_2)}{A \cdot \cos(\varphi_1) + B \cdot \cos(\varphi_2)} .[12] \tag{3-12}$$

Vzorce vztahující se k rovnici (3-9) lze odvodit snadno pomocí přičtení hodnoty π/2 k úhlu φ_2 [9]. Při využití vlastností

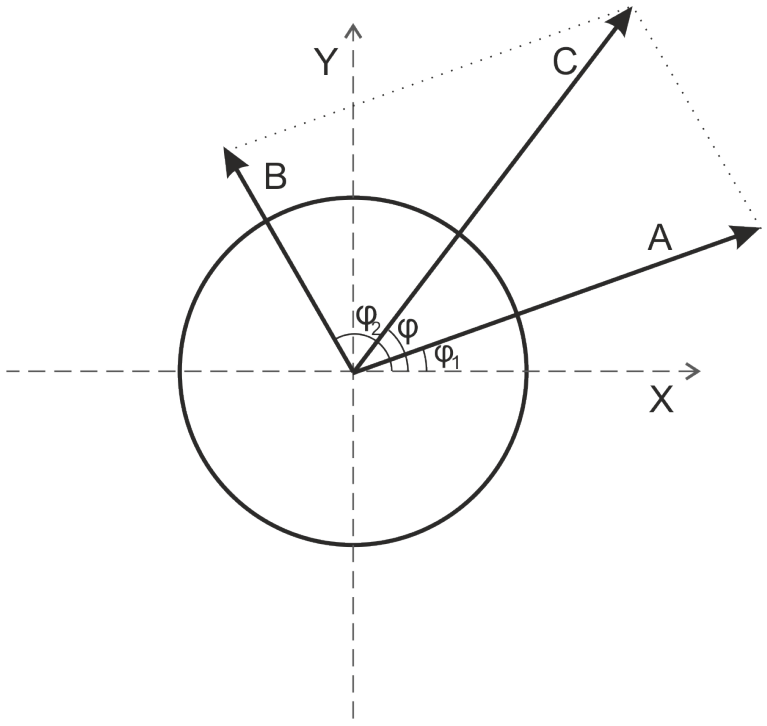
$$\sin\left(x + \frac{\pi}{2}\right) = \cos(x), \tag{3-13}$$

$$\cos\left(x + \frac{\pi}{2}\right) = -\sin(x), \tag{3-14}$$

získáme rovnice

$$C = \sqrt{A^2 + B^2 - 2AB \cdot \sin(\varphi_2 - \varphi_1)} , \tag{3-15}$$

$$\tan(\varphi) = \frac{A \cdot \sin(\varphi_1) + B \cdot \cos(\varphi_2)}{A \cdot \cos(\varphi_1) - B \cdot \sin(\varphi_2)} . \tag{3-16}$$



Obrázek 3.3: Průvodiče harmonických funkcí sinus s posunutou fází

3.4 Popis apletu

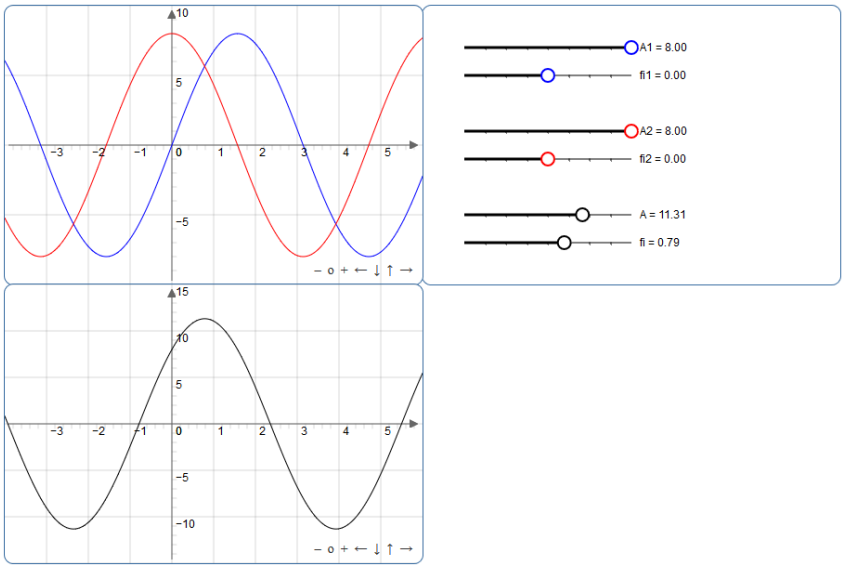
Aplet je funkčně rozdělen na tři samostatně ohraničené sekce (viz Obrázek 3.4). Vpravo nahoře je ovládací sekce. Zde jsou tři dvojice posuvníků, vždy jeden představující modul a druhý fázi harmonické funkce. První a druhá dvojice slouží pro nastavení parametrů vstupních funkcí. Moduly lze nastavit v rozmezí 0 až 8, fáze pak v rozmezí $-\pi$ až π. Jednotlivé dvojice jsou barevně odlišeny. Poslední dvojice je pouze zobrazovací. Ukazuje velikost modulu a fáze výsledné funkce.

Možnost udělat i tyto posuvníky nastavitelné uživatelem s tím, že by se podle nich zpětně dopočítávaly hodnoty ostatních posuvníků, byla zamítnuta, protože takovýto předpis funkce by byl nejednoznačný. To lze dokázat nepřímým důkazem: Pakliže lze stejných výstupních hodnot dosáhnout alespoň dvěma různými kombinacemi hodnot vstupních, předpis je nejednoznačný. Příklad takovýchto kombinací ukazuje následující rovnice:

$$1\cdot\sin(x+0)+1\cdot\cos(x+0)=1\cdot\sin\left(x+\frac{\pi}{2}\right)+1\cdot\cos\left(x-\frac{\pi}{2}\right). \tag{3-17}$$

V sekci vlevo nahoře se vykreslují grafy vstupních signálů. Funkce sinus je vykreslena modře, funkce kosinus červeně. Tyto barvy odpovídají barvám ovládacích posuvníků.

Vlevo dole je sekce s grafem výstupního signálu. Ten je zobrazen černě. Každý graf má vpravo dole svou sadu navigačních tlačítek, které umožňují zoom a pohyb v rovině grafu.



Obrázek 3.4: Rozhraní apletu Součet sinus a kosinus

3.5 Implementace

Kód je rozdělen do tří částí, odpovídajících jednotlivým sekcím apletu. Každá část začíná vytvořením rámce. Ten v případě řídící sekce slouží pouze jako ohraničení prostoru, u ostatních sekcí má přímo funkci grafu.

Do rámce řídící sekce `board0` jsou umístěny posuvníky `slider` funkcí `create`, například

```
A1 = board0.create('slider',[[1,8.5],[5,8.5],[0,8,8]], {name:'A1',
strokeColor:'blue' });
```

která umístí posuvník pro modul funkce sinus mezi souřadnice (1; 8,5) a (5; 8,5) s rozsahem hodnot 0 až 8 a počáteční hodnotou 8.

Dále tato sekce obsahuje funkci `moving`, která je volána vždy při pohybu kteréhokoliv ze vstupních posuvníků a má na starost obnovení spočítaných hodnot a jejich nastavení na výstupních posuvnících. Poslední položkou jsou funkce `getA()` a `getFi()`, které podle rovnic (3-15) a (3-16) počítají koeficienty pro výstupní funkci.

Rovnice pro výpočet fáze ale představuje pro implementaci určitý problém. Funkce Arcus Tangens, která je v ní použita má periodu π. Pokud se s ní počítá fáze harmonické funkce s periodou 2π, výsledky budou správné pouze v rozmezí

$$\left(-\frac{\pi}{2}+2k\pi;\frac{\pi}{2}+2k\pi\right). \tag{3-18}$$

To bylo vyřešeno pomocí tzv. dvouargumentové funkce arcus tangens, která je pro tyto účely definovaná s periodou 2π.

Rámec `board1` dědí funkce a objekty řídícího rámce. Na jejich základě jsou funkcemi

```
board1.create('functiongraph', [function(x){return
  A1.Value()*Math.sin(x+fi1.Value());},], {strokeColor:'blue'});
board1.create('functiongraph', [function(x){return
  A2.Value()*Math.cos(x+fi2.Value());},], {strokeColor:'red'});
```

vytvořeny grafy vstupních funkcí.

Stejným způsobem je ve třetím rámci board2 vykreslen graf výstupní funkce pomocí

```
board2.create('functiongraph', [function(x){return getA(A1, A2,
  fi1, fi2)*Math.sin(x + getFi(A1, A2, fi1, fi2));},],
  {strokeColor:'black'} );.
```

4 APLET FILTRACE SIGNÁLU

4.1 Spektra

Dnešní svět je z velké části postaven na všemožných sdělovacích a komunikačních prostředcích. Všechny tyto prostředky ke své činnosti nevyhnutelně potřebují signály, ať už se jedná o signály zvukové, obrazové nebo signály nesoucí textovou či jinou informaci.

Abychom s těmito signály mohli pracovat, potřebujeme nějak popsat jejich vlastnosti. K tomu máme dvě možnosti. Zaprvé, můžeme signál zkoumat v časové oblasti, tak jak jej zaznamenáváme. Výhodou je, že k tomu není potřeba signál nijak složitě zpracovávat, pouze si zobrazíme jeho graf. Takto však můžeme zkoumat pouze některé parametry, typicky maximální a efektivní hodnotu, periodicitu a základní frekvenci. Nemáme ale možnost zjistit vlastnosti jednotlivých složek signálu, díky čemuž jej například nemůžeme nijak efektivně komprimovat.

Proto se dnes často používá jiná metoda. Ta je postavená na teorii, že každý signál jakéhokoliv průběhu lze rozložit na větší či menší počet harmonických funkcí. Těchto jednotlivých složek může být až nekonečné množství a dohromady vytvářejí spojité frekvenční spektrum. Pokud si zobrazíme toto spektrum jako závislost velikosti modulů harmonických složek na frekvenci, získáme kompletní přehled o vlastnostech signálu, na jejichž základě jej lze dále zpracovávat, například efektivně komprimovat množství dat audiosignálu na základě znalostí psychoakustických vlastností lidského ucha [13]. Další výhodou je, že v lineárních systémech platí pro jednotlivé složky princip superpozice a lze tak snáze odhadnout vliv složek na celkovou odezvu systému. Operaci zobrazení spektra signálu označujeme jako harmonická analýza [10].

4.2 Harmonická analýza

Za otce harmonické analýzy je považován francouzský fyzik a matematik Jean Baptiste Fourier (1768–1830). Na základě jeho prací byly definovány Fourierovy řady (nebo také Fourierův rozvoj) a Fourierova transformace, které na jeho počest nesou jeho jméno [10].

Fourierovy řady slouží k rozkladu periodického signálu na dílčí složky. Pro funkci $f(x)$ s periodou 2π, kdy f je po částech spojitá, se FR vyjádří jako

$$\frac{a_0}{2} + \sum_{k=1}^{\infty} \left[a_k \cdot \cos(k\omega_0 x) + b_k \cdot \sin(k\omega_0 x) \right], \tag{4-1}$$

kde

$$a_k = \frac{2}{T_0}\int_0^{T_0} f(x)\cos(k\omega_0 x)\mathrm{d}x \tag{4-2}$$

$$b_k = \frac{2}{T_0}\int_0^{T_0} f(x)\sin(k\omega_0 x)\mathrm{d}x \tag{4-3}$$

jsou Fourierovy koeficienty pro k-té harmonické složky, tedy násobky základní frekvence $\omega_0 x$. T_0 je základní perioda signálu. Z toho vyplývá, že všechny složky vždy budou násobkem základní frekvence[9].

S využitím vztahu (3-5) můžeme odvodit tvar Fourierových řad, který každé složce přiřazuje pouze jeden koeficient C_k, který odpovídá modulu k-té složky a fázový posun složky φ_k:

$$\frac{a_0}{2} + \sum_{k=1}^{\infty} C_k \sin(k\omega_0 x + \varphi_k). \tag{4-4}$$

Koeficient a_0 udává velikost stejnosměrné složky signálu.

V případě, že má signál skokové změny, tak pro dokonalou aproximaci tvaru signálu je třeba nekonečný počet harmonických složek. V případě konečného počtu složek vznikají v okolí zlomu překmity. Tomuto se říká **Gibbsův jev**.

Fourierova transformace umožňuje získat i koeficienty neperiodického signálu. Její výpočet je ale výrazně náročnější. Pro funkci $s(t)$, která je absolutně integrovatelná v intervalu $(-\infty;\infty)$, je FT definována rovnicí

$$S(\omega) = \int_{-\infty}^{\infty} s(t)\mathrm{e}^{-\mathrm{j}\omega t}\mathrm{d}t, \tag{4-5}$$

kde ω je úhlový kmitočet, $\omega = 2\pi f$. $S(\omega)$ je Fourierovým obrazem funkce $s(t)$ [10].

FT nejčastěji se nejčastěji využívá ke zpracování diskrétního (navzorkovaného) signálu. Pak je potřeba znát i její vyjádření v diskrétní oblasti:

$$S(\mathrm{e}^{\mathrm{j}\omega}) = \sum_{n=-\infty}^{\infty} s[n]\mathrm{e}^{-\mathrm{j}\omega n}. \tag{4-6}$$

Pro výpočet pomocí počítače je třeba mít diskrétní nejen vstupní signál $s[n]$, ale i výstup, spektrum $S(\mathrm{e}^{\mathrm{j}\omega})$ [14].

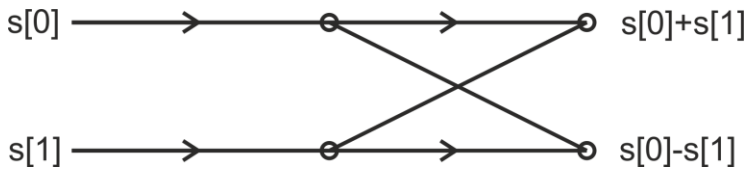
4.2.1 Rychlá Fourierova transformace

Jako rychlá Fourierova transformace (anglicky Fast Fourier Transform) se označuje soubor algoritmů, které různými způsoby zefektivňují výpočet FT. Nejznámější a zároveň nejstarší je algoritmus z roku 1965, jehož autory jsou James W. Cooley a John W. Tukey. Ten využívá zjednodušení některých členů na základě periodicity funkce e^{jx}, např.:

$$e^{-j\frac{3\pi}{2}} = -e^{-j\frac{\pi}{2}}, \tag{4-7}$$

$$e^{-j3\pi} = e^{-j\pi} = -1. \tag{4-8}$$

Tím dojde k rozdělení posloupnosti na sudou a lichou část. U toho dojde mimo jiné ke změně pořadí koeficientů. Tak se pokračuje, dokud nezbude pouze jeden komplexní činitel. Následně se provede operace postupného dvoubodového sčítání posloupností. Tato operace se někdy označuje jako „motýlek“ z důvodu podobnosti jejího zápisu signálových toků s motýlími křídly (viz Obrázek 4.1) Zde je pro efektivnost algoritmu důležité, aby počet vzorků odpovídal mocnině dvojky [10][14].



Obrázek 4.1: Dvoubodové sčítání signálových toků („motýlek“)

4.3 Kmitočtové filtry

Kmitočtové filtry jsou lineární systémy, sloužící ke změně spektra signálu. Mohou být analogové, realizované pomocí elektrických obvodů a pracující se spojitým signálem, nebo číslicové (digitální), pracující s diskrétním signálem. Vlastnosti filtrů jsou charakterizovány kmitočtovou charakteristikou, tedy závislostí velikosti modulu harmonických složek a jejích fáze na frekvenci [6].

4.3.1 Typy filtrů

Filtry se rozdělují podle kmitočtové charakteristiky. Nejběžnějšími typy jsou horní, dolní a pásmová propust a pásmová zádrž.

Horní propust je definovaná mezním kmitočtem f_m. Složky vyšší než f_m propouští, na f_m má přenos -3 dB a směrem k nižším složkám s určitou strmostí klesá

až k nulovému přenosu. Ideální HP má nekonečnou strmost. **Dolní propust** je opakem Horní propusti. Propouští pásmo pod mezním kmitočtem a pásmo nad ním potlačuje.

Pásmová propust může být definovaná jako kombinace DP a HP, z nichž každá má svůj mezní kmitočet. PP pak propouští pouze složky ležící mezi těmito kmitočty. Jiná možnost definice je pomocí středního kmitočtu a šířky pásma. **Pásmová zádrž** je opakem PP, tedy propouští vše, kromě frekvencí ležících kolem středního kmitočtu.

Speciálním případem je fázovací článek, jehož modulová charakteristika je konstantní a ovlivňuje pouze fázi [14].

Mnoho filtrů nevyhovuje této klasifikaci, případně jsou jejich kombinací. Tyto filtry označujeme jako kmitočtové korektory (ekvalizéry). Příkladem mohou být filtry shelving a peak, které mění zesílení části spektra na jinou než nulovou úroveň, zbytek spektra nechávají neovlivněný [7].

4.3.2 Návrh filtrů

Analogové filtry se navrhují z reálných součástek tak, aby se přiblížily požadované ideální přenosové funkci. U filtrů prvního řádu bude strmost přenosové funkce 20dB na dekádu. To pro mnoho aplikací není dostačující. Se zvyšujícím se řádem filtrů a z toho vyplývající zvyšující se strmostí výrazně stoupá jeho složitost a tedy i cena [6].

Oproti tomu v digitální oblasti není problém navrhnout ideální filtr s nekonečnou strmostí. K tomu se používá u filtrů typu HP typicky funkce jednotkového skoku $u(x)$:

$$u(x - f_m) = 1 \text{ pro } x \geq 0, \tag{4-9}$$

$$u(x - f_m) = 0 \text{ pro } x < 0, \tag{4-10}$$

kde f_m je mezní kmitočet filtru. Pro DP se použije funkce s opačnou polaritou.

Filtr typu PP se realizuje sériovou kombinací dolní a horní propusti, tedy součinem dvou funkcí jednotkového skoku. PZ vznikne při součtu dvou funkcí jednotkového skoku, přičemž funkce představující DP má nižší mezní kmitočet než funkce představující HP.

4.4 Popis apletu

Aplet se skládá ze tří základních sekcí, navzájem odlišených barvou grafů a posuvníků. (Obrázek 4.2). Vlevo se nachází vstupní sekce označená modře. Zde je možné vybrat jeden ze sedmi typů signálu. První tři signály jsou generované

oscilátorem. Jedná se o sinusovou, pilovou a obdélníkovou funkci, kterým je možné pomocí posuvníku nastavit frekvenci v rozmezí 50 Hz až 11 kHz. Další čtyři signály jsou nahrány jako soubory typu wav. Jsou jimi zvuk malého bubnu, zvuk zkreslené a nezkreslené elektrické kytary a bílý šum.

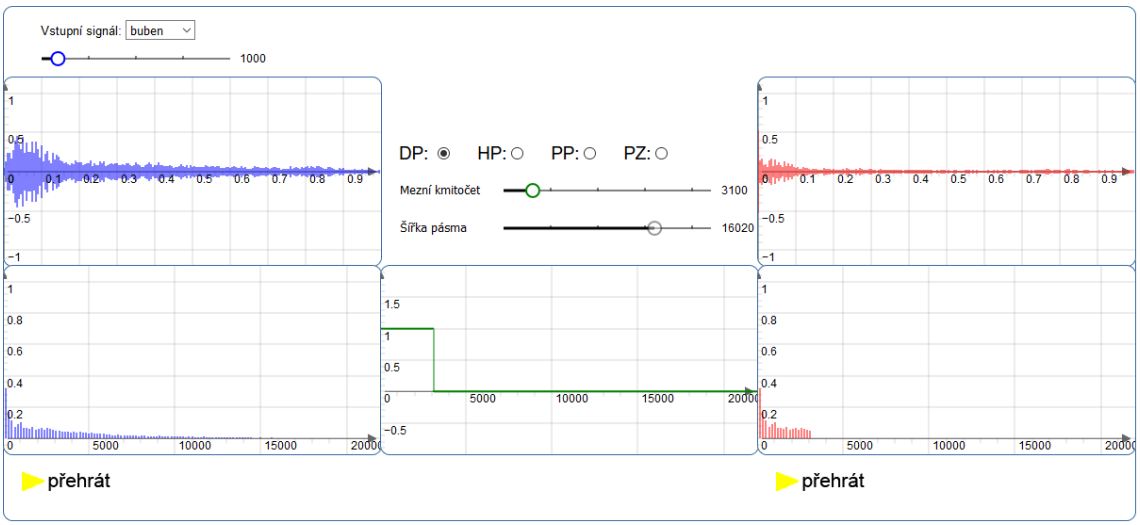
Ve vstupní sekci jsou dále dva grafy zobrazující vybraný signál. První je graf v časové oblasti. V případě že je vybraný generovaný zvuk, zobrazuje jeho průběh v délce 1 ms. Pokud je vybraný zvuk nahraný ze souboru, v grafu se zobrazí waveforma prvních 250 ms. Druhý graf zobrazuje frekvenční modulové spektrum signálu v rozsahu 0 až 22 kHz.

Prostřední sekce představuje filtr. Je možné přepínat mezi čtyřmi typy filtru označených zkratkami českých názvů: dolní propustí (DP), horní propustí (HP), pásmovou propustí (PP) a pásmovou zádrží (PZ). Dále jsou zde dva posuvníky. U prvních dvou typů filtru slouží první posuvník k nastavení mezního kmitočtu, druhý posuvník je neaktivní. U druhých dvou typů filtru první posuvník nastavuje střední kmitočet a druhý posuvník slouží k nastavení šířky pásma. Rozsah je vždy 0 až 22 kHz.

Důležitou součástí sekce filtru je graf, který zobrazuje přenosovou funkci filtru. Jeho rozsah je rovněž 0 až 22 kHz. Prvky sekce filtru mají zelenou barvu.

Výstupní sekce, stejně jako vstupní sekce obsahuje dva grafy, tentokrát vykreslené červenou barvou. V nich se opět zobrazuje časový průběh a spektrum, tentokrát ovšem již po aplikaci filtru.

Součástí vstupní i výstupní sekce jsou tlačítka pro přehrání signálu v dané sekci. Aktivace přehrávače v jedné sekci vypne přehrávání v druhé.



Obrázek 4.2: Rozhraní apletu Filtrace signálu

4.5 Implementace

V tomto apletu bylo nutné kromě obvyklého rozdělení javascript kódu do sekcí podle rámců přidat ještě jednu sekci, která obsahuje funkce pro práci se zvukem. Pro implementaci těchto funkcí nebyla zvolená žádná externí knihovna, nýbrž rozhraní Web Audio API nativně podporované moderními webovými prohlížeči jako přirozená součást javascriptu.

Jeho podpora mezi prohlížeči užívanými běžnou veřejností je v době psaní práce pouze cca 80%, podstatnou část nepodporovaných platforem ale tvoří prohlížeče pro mobilní telefony, takže je jeho použití dnes již plně legitimní. [15]

Vzhledem k rozdílnému způsobu práce se signály nahranými ze souboru a vytvořenými oscilátorem bylo nutné část funkcí pro práci se zvukem vytvořit ve dvou verzích. Společná je inicializace všech součástí řetězce pracujících v reálném čase, funkce pro spuštění a zastavení přehrávání a funkce pro propojení jednotlivých modulů Web Audio API tak, jak je podle aktuálního nastavení potřeba. Pro generované signály je pak použita funkce pro nastavení parametrů oscilátoru, zatímco pro samplované zvuky je implementována funkce, která je při spuštění předehraje do pole bufferu, které jsou v případě spuštění kopírovány do přehrávacího bufferu.

Web Audio API obsahuje přednastavené filtry 2. řádu. Nemá však žádnou přirozenou možnost, jak filtrovat signál ve spektrální oblasti. Jako řešení se nabízelo implementovat strukturu filtru do modulu typu `ScriptProcessorNode` určeného pro zpracování přehrávaného signálu. Výkon poskytnutý prohlížečem však nebyl dostatečný a přehrávaný zvuk i při malém okně pro FFT (256 samplů) nebyl přehráván plynule.

Z toho důvodu byl pro přehrávání zvuku tento filtr nahrazen dvěma kaskádami filtrů 2. řádu, jednou představující horní propust a druhou představující dolní propust. Každá kaskáda obsahuje šest filtrů 2. řádu, což dohromady poskytuje strmost 120 dB/dekádu. Taková strmost ještě není na daném měřítku zcela nepostřehnutelná, ovšem v praxi ji můžeme zanedbat.

Při přehrávání je zdroj signálu (buffer nebo `oscilatorNode`) zapojen do kaskády předvytvořených filtrů typu `BiquadFilter` poskládaných tak, aby dohromady vytvořily požadovaný typ filtru. Poslední součástí řetězce je `AudioContext.destination`, které tvoří výstup do zvukového zařízení. Podle toho jestli je přehrávač spuštěný ve vstupní nebo výstupní sekci je možné jej zapojit před filtr nebo za něj.

To vše se ovšem děje v reálném čase. Oproti tomu grafy bylo třeba vykreslovat co nejrychleji. Proto jsou implementovány dvě funkce, které zcela nezávisle na funkcích

pro zpracování v reálném čase provedou FFT, čímž získají spektrum pro vstupní graf, na to aplikují filtr, čímž získají spektrum pro výstupní graf, a aplikací inverzní FFT získají tvar průběhu výstupního signálu. Varianta pro generované signály se liší sto dvaceti pětinásobným nadvzorkováním vycházejícím z měřítka grafů, které činí 1 ms. Okno pro FFT tvoří prvních 8192 vzorků signálu.

Waveformy a spektra jsou do grafů vykresleny jako pole čar (u waveforem 200 čar, u spektra 128), kterým jsou v případě změny přesunuty na odpovídající pozice. Obojí ukazuje efektivní hodnotu signálu.

Javascript obecně není jazyk optimalizovaný pro vysoký výkon, spíše pro jednoduchost práce a interakci s HTML. Tento aplet díky tomu již naráží na hranice toho, co je na dnešním hardwaru možné v reálném čase použít. Filtrace v reálném čase musela být nahrazena jiným řešením. Dále například překreslování grafů není plynulé tak jako u původní verze apletu v jazyce Java, nýbrž reaguje na událost `mouseUp`, tedy na puštění tlačítka myši a dochází k němu s viditelným zpožděním.

Jedna z variant návrhu navíc počítala s vykreslováním spekter v reálném čase při přehrávání zvuku. Největší použitelné okno transformace, u kterého nedocházelo k zasekávání přehrávání nebo dokonce pádu prohlížeče, činilo 128 vzorků. To nebylo pro kvalitní zobrazení dostatečné, tudíž tato varianta byla také zamítnuta.

5 APLET VÝZNAM FÁZE U AUDIOSIGNÁLU

5.1 Fáze signálu

Jak již bylo řečeno v dřívějších kapitolách, lze jakýkoliv signál pomocí Fourierovy transformace rozložit na jednotlivé spektrální složky. Výstupem FT jsou pak koeficienty a_k a b_k pro k-té složky, získané z rovnic (4-2) a (4-3), které představují velikost kosinusové (a_k) a sinusové (b_k) složky na této frekvenci. Tyto dvě složky lze pak převést na jedinou sinusovku (případně i kosinusovku) podle vzorce (4-4), určenou modulem c_k a fází φ_k. Pokud se pro tyto koeficienty sestaví zvlášť závislosti na frekvenci, výsledné grafy budou představovat modulové a fázové spektrum.

Pro přesnou rekonstrukci jakéhokoliv signálu je nezbytné znát hodnoty obou spekter. Výsledný tvar signálu dotváří každá jednotlivá složka a jakékoliv posunutí fáze kterékoliv složky se na signálu vždy více či méně projeví. V případě rozsáhlejších změn fází složek bude výsledný signál doslova k nepoznání od původního.

V oblasti audiosignálu je situace trochu jiná. Lidské ucho nevyhodnocuje zvuk v časové oblasti, tak jako běžně používané A-D převodníky, nýbrž podle spektra zaznamenaného signálu. Souvisí to s fyziologickou stavbou ucha, kdy receptorové buňky jsou rozloženy postupně po délce hlemýždě ve vnitřním uchu. Každá frekvence má rezonanční bod v jiné části hlemýždě (směrem dovnitř od nejvyšších frekvencí po nejnižší) a tím pádem na každou receptorovou buňku připadá jen velice úzká část spektra. To mimo jiné přispívá k lepší srozumitelnosti a zpracovatelnosti zvuku. Zároveň to také vede k jevům zvaným frekvenční maskování, které jsou využívány pro dnes hojně používané algoritmy ztrátové komprese datového toku audio souborů [13].

Pro většinu případů práce se zvukovými signály je tedy informace o fázích jednotlivých složek nepotřebná. To dokládá i fakt, že ve většině profesionálních DAW softwarů (Steinberg Cubase, Avid Pro Tools…) jsou FFT analyzátory ukazující pouze amplitudové spektrum.

Není ale pravda, že lidské ucho nevnímá fázi zvuku žádným způsobem. Velmi důležitá je fáze pro binaurální slyšení. Pomocí fázového rozdílu mezi signálem v levém a v pravém uchu lidský mozek určuje směr šíření zvuku a tedy i lokalizuje jeho zdroj. Pro hluboké složky slyšitelného spektra (většinou se uvádí do kmitočtu 400Hz, někdy i 700Hz) je tato tzv. interaruální fázová diference jediným zdrojem lokalizace směru, protože pro vlny s tak velkou vlnovou délkou se neuplatňuje akustický stín hlavy a na něm založená metoda interaruální intenzitní diference.

Zajímavým faktem v oblasti vyhodnocování směru signálu podle fáze je tzv. Haasův jev nazývaný také jev priority, který říká, že vlna, jež dorazí do jednoho z uší jako první je vyhodnocována prioritně i v případě, že vlna v druhém uchu bude hlasitější a to až do rozdílu intenzity 7–10dB [13].

5.2 Spektrum obdélníkového signálu

Při přenášení digitálního signálu je často nezbytné přenést nespojitou číselnou informaci pomocí spojité veličiny, například pomocí napětí. Tehdy, a v mnoha dalších případech se často používá impulsní signál, který je tvořen pravoúhlými impulzy o určité výšce D a délce ϑ. Periodizací takového signálu získáme signál obdélníkový.

Ten je charakteristický periodickým střídáním části s nenulovou hodnotou o velikosti D a s nulovou hodnotou. Pro jeho popis jsou nezbytné další veličiny – perioda T_1, popřípadě frekvence $f_1 = 1/T_1$, a střída DCL, která vyjadřuje poměr mezi délkou nenulové části a délkou nulové části v jedné periodě:

$$DCL = \frac{\vartheta}{T_1 - \vartheta}, \tag{5-1}$$

uvádí se jako poměr dvou čísel (například 1:2).

Pokud takovýto signál dosadíme do rovnice pro Fourierovu řadu (4-4), dostaneme úpravou tvar pro k-tý koeficient modulového spektra

$$c_k = \frac{D\vartheta}{T_1} \cdot \frac{\sin\left(k\omega_1 \frac{\vartheta}{2}\right)}{k\omega_1 \frac{\vartheta}{2}}, \tag{5-2}$$

když platí, že základní úhlová rychlost $\omega = 2\pi f_1$. Pro zjednodušení tohoto vzorce je vhodné využít funkci **sinus cardinalis**, která je definovaná jako

$$\operatorname{sinc}(x) = \frac{\sin(x)}{x} \text{ pro } x \neq 0, \tag{5-3}$$

$$\operatorname{sinc}(x) = 1 \text{ pro } x = 0. \tag{5-4}$$

Výsledný vzorec bude vypadat takto

$$c_k = \frac{D\vartheta}{T_1} \cdot \operatorname{sinc}\left(k\omega_1 \frac{\vartheta}{2}\right). \tag{5-5}$$

Tvar funkce sinc vytváří střídavě kladné a střídavě záporné vlny, které se vzdalujíce se od počátku zkracují a zmenšují. Vzhledem k tomu, že moduly v celém spektru potřebujeme kladné, je třeba je vyjádřit jako absolutní hodnotu $|c_k|$. To si ale

žádá úpravu fáze pro složky, jimž byla otočena polarita. K jejich fázi φ_k tedy přičteme $\pm\pi$ tak, aby platilo $(-1) = \mathrm{e}^{\pm \mathrm{j}\pi}$. Výsledné modulové spektrum je pak tvořeno jednotlivými harmonickými složkami k základní frekvenci s obálkou tvořenou funkcí sinc. [10]

Toto spektrum je charakteristické nulovými body. Jejich poloha ve spektru vychází ze střídy a v případě střídy definované poměrem celých čísel připadá na násobky základní frekvence. Tuto hodnotu kmitočtu ω_z pro *i*-tý nulový bod lze spočítat z rovnice

$$\sin\left(\omega_z \tfrac{\vartheta}{2}\right) = i\pi\,, \tag{5-6}$$

po úpravě pak

$$\omega_z = i\frac{2\pi}{\vartheta} = i\frac{\omega_1 T_1}{\vartheta}\,. \tag{5-7}$$

5.3 Popis apletu

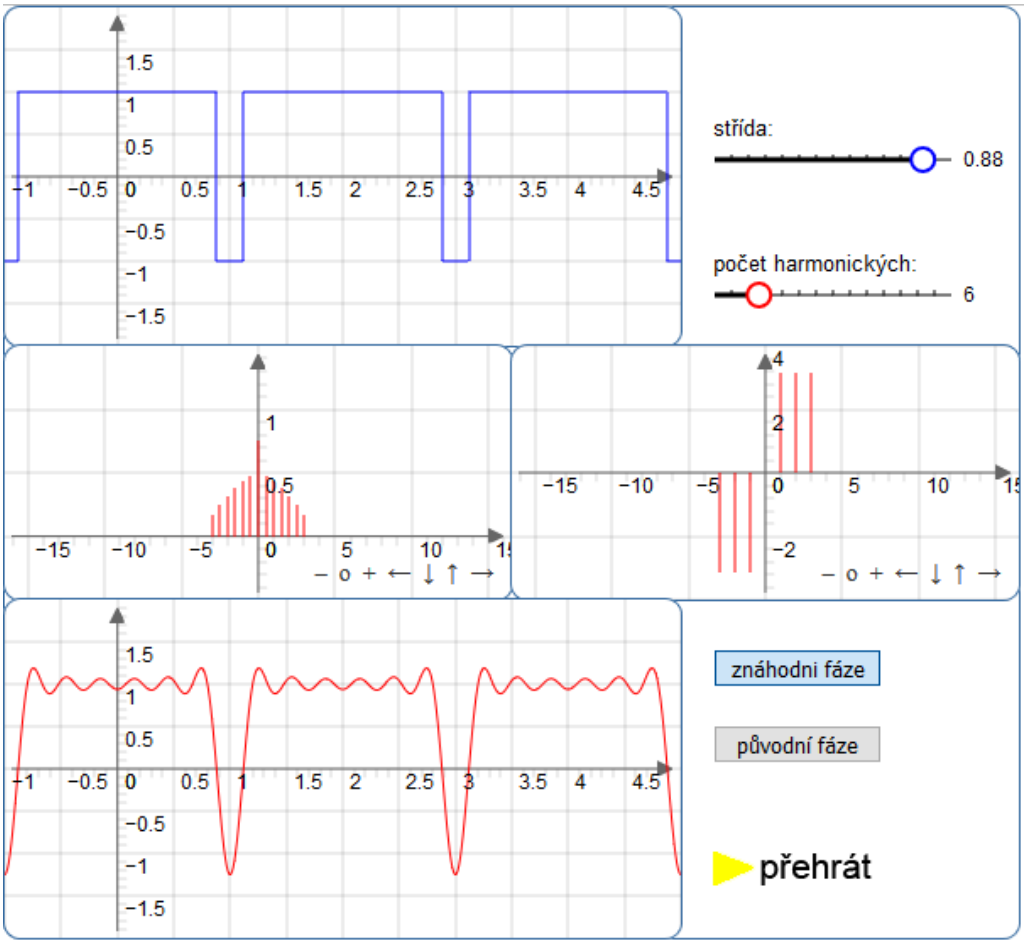
Tento aplet je členěn horizontálně na tři sekce (Obrázek 5.1). První sekce, značená modře, zobrazuje vstupní obdélníkový signál o základní periodě 2 ms. K němu se váže posuvník pro nastavení střídy. Obdélníkový signál byl vybrán kvůli relativní jednoduchosti odvození vzorců pro spektrum a názornosti takové ukázky.

Ve druhé sekci jsou dva grafy, které zobrazují modulové a fázové frekvenční spektrum obdélníkového signálu rozloženého pomocí fourierových řad (5-5). Je možné posuvníkem volit počet harmonických složek v rozmezí 0 až 32. Spektra pak mají rozsah $-16{,}5$ kHz až 16,5 kHz, což odpovídá maximálnímu počtu harmonických složek.

Ve spodu se pak nachází výstupní graf, který zobrazuje signál zrekonstruovaný ze spočítaných koeficientů zobrazených na grafech v prostřední části. Tento signál je možné přehrát. Druhá i třetí sekce jsou označeny červenou barvou, aby bylo zřejmé, že spolu úzce souvisí.

Posledním prvkem je tlačítko „znáhodni fáze“, které fázím jednotlivých harmonických složek přiřadí náhodné hodnoty v rozmezí $-\pi$ až π. To zcela zásadně změní tvar rekonstruovaného signály, který již příliš nebude připomínat původní obdélníkový signál. Na přehrávaný zvuk ovšem tato změna nemá vliv.

Tuto změnu je možné vrátit zpět buď stiskem tlačítka „původní fáze“ nebo tahem za posuvník pro změnu střídy.



Obrázek 5.1: Rozhraní apletu Význam fáze audiosignálu

5.4 Implementace

Všechny grafy jsou opět vykresleny pomocí knihovny JSXGraph. Vstupní signál je definován logickou strukturou, která pro každé x vrací buď hodnotu 1, nebo -1. Výstupní graf je vytvořen cyklem, který pro každé x sečte hodnoty všech harmonických funkcí v tomto bodě a poté jej zobrazí do grafu.

Koeficienty modulů i fází jsou definovány v polích, do kterých se při každé změně parametrů vždy znovu spočítají. Z těchto polí jsou pak volány pro vykreslení spekter, výstupního grafu i pro nastavení přehrávače. Vše probíhá pomocí cyklu, pouze 0. složka je vždy zpracována zvlášť, mimo cyklus.

Funkce pro znárodnění fází je realizována pomocí dvou stupňů polí pro koeficienty fáze. Skutečný koeficient obdélníkového signálu se spočítá do pole

`poleFazi[]`, pro vykreslování se ale používá `poleFaziPost[]`, do kterého se buď zkopírují hodnoty z původního pole, nebo se vygenerují nové funkcí:

```
poleFaziPost[i] = Math.PI*(2*Math.random()-1).
```

Pro přehrávání je použita knihovna Gibberish, která umožňují jednoduchou práci s generovanými zvuky jako objekty. Není tedy potřeba se starat o zapojení jednotlivých modulů audio kontextu, jako při použití Web Audio API. Její podpora v prohlížečích ale pravděpodobně není tak široká. Přesné údaje nejsou známé, nicméně při testování se jej podařilo spustit pouze na prohlížečích Firefox verze 52 a Chrome verze 58.

6 APLET ALIASING A JEHO PROJEVY

6.1 Aliasing

Pozorujeme-li projíždějící auta skrze zábradlí či plot se svislýma příčkami, můžeme si všimnout zvláštního jevu – kola automobilu často vypadají, jako že se točí pozpátku nebo jakoby stojí přesto, že se auto pohybuje. Tento jev se nazývá aliasing.

K aliasingu dochází všude tam, kde se při vzorkování snažíme zachytit jevy rychlejší, než je vzorkovací frekvence. K tomu přeneseně dochází i u výše zmíněného příkladu. Mezery mezi příčkami plotu slouží jako jednotlivé vzorky a úhlová frekvence otáčejícího-se kola je výrazně větší, než množství příček, kolem kterých za tu dobu auto projede. Kolo se ani nemusí skutečně otáčet tak velkou rychlostí, většinou mu pomůže jeho symetričnost (např. je li kolo symetrické podél tří os, zdánlivá úhlová rychlost se zvýší šestkrát).

S aliasingem se setkáváme také v elektrotechnice, například při vzorkování audiosignálu. Obecné pravidlo říká, že aby byla navzorkovaná harmonická funkce jednoznačně definovatelná po stránce frekvence a amplitudy, musí být zaznamenána více než dvěma vzorky na jednu periodu. Pokud bude počet vzorků menší, nebude možné tuto harmonickou složku zpětně rekonstruovat. Navíc hodnoty, které do vzorkování vnese, vytvoří parazitní složky na jiné frekvenci a ty negativně ovlivní výsledný signál. Jedná se tedy o formu nelineárního zkreslení signálu.

Zvláštní situace nastane u harmonické složky, jejíž frekvence je přesně dvojnásobná, než vzorkovací frekvence. Její frekvence je ve vzorcích definovaná správně, její amplituda ale může nabýt libovolné hodnoty od nuly až po skutečnou amplitudu. Záleží, do jaké hodnoty se vzorek „trefí“. Dochází zde tedy pouze k lineárnímu zkreslení, nové harmonické složky se netvoří. Hodnotu ale ani tak není možné brát jako správnou.

Na základě těchto poznatků je definován Nyquistův teorém

$$f_{vz} > 2 f_{\max}, \tag{6-1}$$

neboli vzorkovací frekvence musí být více než dvojnásobná, než frekvence největší zaznamenávané složky. [10]

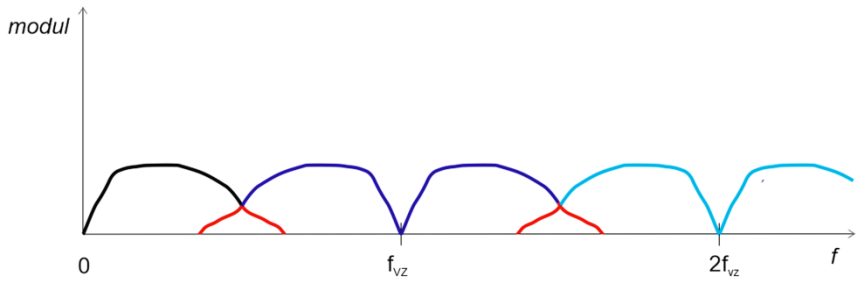
Jediná možnost správného navzorkování spojitého signálu spočívá ve vzorkování signálu, který takovéto složky neobsahuje. Proto se vždy před A-D převodník umísťuje tzv. antialiasingový filtr. Jedná se o filtr typu dolní propust s mezním kmitočtem okolo poloviny vzorkovací frekvence. V praxi je ale nemožné dosáhnout u analogového filtru ideální strmosti. Vzorkovací frekvence se tedy vybírá

s dostatečnou rezervou. Kupříkladu nejběžnější a zároveň nejnižší běžně používaná vzorkovací frekvence u audiosignálu je 44,1 kHz, přičemž za nejvyšší lidským uchem slyšitelnou frekvenci se obecně považuje 20 kHz, tedy méně než polovina. [13]

6.1.1 Spektrum vzorkovaného signálu

U navzorkovaného signálu je potřeba spektrum počítat pomocí diskrétní Fourierovy transformace (4-6). Z jejích vlastností vyplývá, že spektrum vzorkovaného signálu je nejenom symetrické kolem osy *y*, navíc se periodicky opakuje s periodou rovnou vzorkovacímu kmitočtu.

Z toho lze odhadnout, že při překročení Nyquistova teorému se spektra v jednotlivých periodách navzájem protnou a ovlivní (Obrázek 6.1). To vysvětluje podstatu aliasingu.



Obrázek 6.1: Překrytí spekter při aliasingu

6.2 Popis apletu

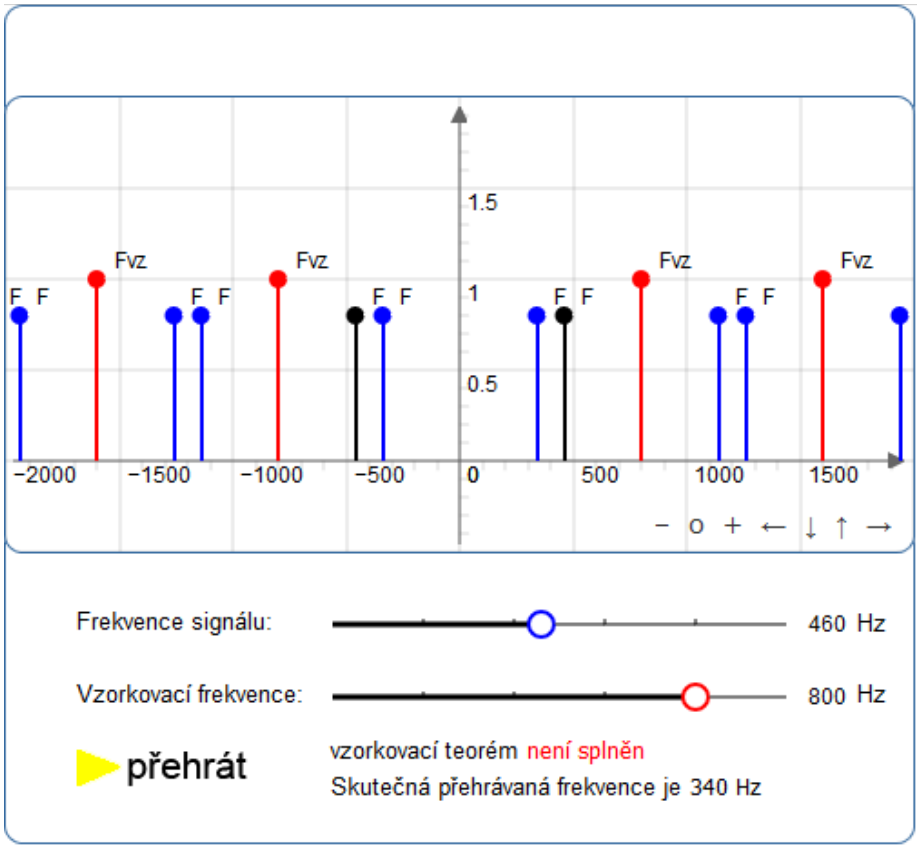
Tento aplet sestává pouze z jedné sekce (Obrázek 6.2). Vstupním signálem je sinus, jehož frekvenci je možné nastavit posuvníkem v rozmezí 0 až 1000 Hz.

Dalším posuvníkem je možné nastavit vzorkovací frekvenci, tentokrát v rozmezí 200 až 1000 Hz. To bylo nutné, protože při nižší vzorkovací frekvenci je již třeba počítat s příliš velkým množstvím spektrálních čar, což často vedlo k zaseknutí prohlížeče. Rozsahy obou posuvníků jsou ale synchronizované (oba v rozmezí 0 až 1000 Hz), pouze u druhého z nich při překročení spodní hranice 200 Hz je jezdec na tuto hodnotu vrácen.

Ústředním bodem apletu je jediný graf, zobrazující spektrální čáry vstupního signálu navzorkovaného se zvolenou vzorkovací frekvencí. Jeho rozsah činí -2000 až 2000 Hz. Červeně jsou zobrazeny čáry znázorňující vzorkovací frekvenci a její násobky (tomu odpovídá červená barva jezdce posuvníku pro nastavení

vzorkovací frekvence). Modré čáry pak představují frekvenci navzorkovaného sinusového vstupního signálu a jeho kopie, periodicky se opakující okolo každého násobku vzorkovací frekvence. Posuvník pro nastavení frekvence signálu je také odlišen modře. Jedinou výjimkou jsou čáry představující skutečnou frekvenci signálu. Ty jsou znázorněny černě.

Ve spodní části je textem zobrazeno, zdali je nebo není dodržen Nyquistův vzorkovací teorém (6-1). Zobrazuje se zde také frekvence, která skutečně odpovídá přehrávanému signálu. Ta v případě, že je teorém splněn, odpovídá vstupnímu signálu. V opačném případě je nižší a odpovídá složce spektra, která je nejblíže k ose y. Posledním prvkem je tlačítko pro ovládání přehrávače.



Obrázek 6.2: Rozhraní apletu Aliasing a jeho projevy

6.3 Implementace

Implementace tohoto apletu opět využívá knihovnu JSXGraph pro vykreslení grafů a Gibberish pro přehrání zvuku.

Spektrální čáry jsou při spuštění inicializovány do polí. Je jich vytvořeno tolik, aby to přesně pokrylo rozsah grafu při nejmenší dostupné vzorkovací frekvenci (ta byla zvolená 200 Hz). Jejich poloha se pak počítá vždy jako $(i \cdot Fvz) \pm F$.

Důležitá je funkce, která kontroluje splnění teorému na základě zadaných vstupních hodnot. Pokud dodržen není, tak spočítá správnou hodnotu jako hodnotu spektrální čáry nejblíže osy y. Tato hodnota se pak zobrazí do HTML input objektu, který plynule navazuje na text ve spodní části. Kromě toho, pokud je aktivní přehrávač tak je touto funkcí spočítána frekvence přehrávaného sinusu.

7 APLET LINEÁRNÍ A NELINEÁRNÍ SYSTÉMY

7.1 Systémy

Jako systém se obecně v teorii signálů označuje souhrn operací, které ovlivňují spektrum signálu, na nějž se aplikují. Se systémy se lze při práci se signálem setkat prakticky kdekoliv. Může se jednat o jakýkoliv filtr nebo efekt, u kterého je úprava spektra signálu jeho hlavním účelem. Za systémy je ale třeba považovat také např. jakékoliv reálné analogové součástky. Dokonce i stíněný a dobře umístěný profesionálně zpracovaný vodič bude mít na spektrum signálu vliv, jakkoliv bude zanedbatelný. Při návrhu zařízení, u kterých je kladen důraz na malé zkreslení je potřeba uvažovat i tyto jevy.

Jako zkreslení se označuje změna, kterou průchod signálu systémem způsobí. Můžeme jej posuzovat ve dvou oblastech, časové a frekvenční.

V časové oblasti zkreslení pozorujeme jako změnu tvaru signálu. Je možné jej popsat pomocí závislosti velikosti výstupního signálu na vstupním, označované jako přenosová křivka. Někdy se nazývá také přenosová funkce signálu, toto označení ale není šťastné vzhledem k možnosti záměny s přenosovou funkcí LTI systému (viz kapitola 7.4). S tímto zobrazením se lze nejčastěji setkat při úpravě dynamiky signálu, například u kompresoru nebo expanzeru audiosignálu, případně u nástroje „křivky“ pro úpravu jasu fotografie. Tato závislost může být různá pro různé frekvence.

Ve frekvenční oblasti lze posuzovat vliv sytému na jednotlivé harmonické složky. Zobrazuje se jako závislost rozdílu velikosti složky ve vstupním a výstupním signálu na frekvenci. Toto zobrazení je běžné a velice názorné u ekvalizačních filtrů. [10]

7.2 Lineární systémy

Základní dělení systémů spočívá v rozdělení na lineární a nelineární systémy. Pro lineární systémy platí princip superpozice. Jeho podstatou je, že nezáleží na pořadí, v jakém se provedou jednotlivé operace. Pokud tedy necháme projít systémem dva signály a poté je sečteme, dostaneme stejný výsledek, jako kdybychom je nejprve sečetli a teprve poté nechali projít systémem. Matematicky vyjádřeno

$$f(a+b)=f(a)+f(b)\,. \tag{7-1}$$

Kromě toho musí platit také princip homogenity:

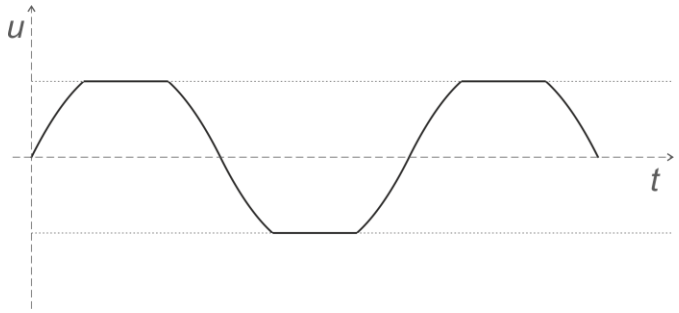
$$ax_1(t) = ay_1(t). \tag{7-2}$$

U takového systému nevznikají žádné nové harmonické složky, dochází pouze ke změně velikostí a fází harmonických složek, které původní signál obsahoval. Přenosové křivky všech frekvenčních složek vždy budou mít tvar přímky procházející středem. [10][14]

7.3 Nelineární systémy

Nelineární systémy lze zjednodušeně shrnout jako všechny, které ve spektru signálu vytvářejí nové harmonické složky. Typicky se jedná o polovodičové součástky, jako dioda nebo tranzistor, dodnes velice vyhledávané mezi kytaristy jsou pak elektronky a z nich postavené zesilovače, které mají tendenci generovat spíše sudé harmonické frekvence a tedy líbivější zkreslení. Typickým příkladem kytarového nelineárního efektu je overdrive nebo distortion.

Extrémním případem je jev zvaný limitace, který uřeže ze signálu vše, co v časové oblasti překročí určitou hranici (Obrázek 7.1). Tuto hranici nazýváme Treshold. V nelineárních systémech neplatí princip superpozice. Záleží tedy na pořadí jednotlivých systémů. [10][16]



Obrázek 7.1: Limitace harmonické funkce

7.4 Časově invariantní systémy

Časově invariantní systém je takový, jehož vlastnosti se nemění s časem. Někdy bývá označován také jako neparametrický systém. V reálném světě je vždy přítomno příliš mnoho působících vlivů. S časově invariantními systémy se zde tedy obvykle nesetkáme. Velice často se ale využívají jako zjednodušené modely. Ke slovu se pak dostávají u diskrétního zpracování signálu. V digitální oblasti jejich implementace

nepředstavuje žádný větší problém. Zde je spíše problém opačný – velice náročná implementace časově proměnných systému, které by představovaly věrnou simulaci reálných hudebních nástrojů.

Z matematického hlediska jsou nejvýhodnější **lineární časově invariantní systémy**, zkráceně LTI (Linear Time Invariant). Ty lze totiž velice jednoduše popsat tzv. přenosovou funkcí

$$H(z)=\frac{Y(z)}{X(z)}, \tag{7-2}$$

kde $X(z)$ představuje obraz vstupního signálu v oblasti Z transformace a $Y(z)$ je obraz výstupního signálu. [10][16]

7.5 Popis apletu

Ústředním bodem apletu je schéma znázorňující zapojení jeho jednotlivých částí (Obrázek 7.2). Od dvou vstupních signálů, znázorněných modrým a zeleným bodem, pokračuje přes první systém, poté dojde ke sloučení signálů a na konci je umístěn druhý systém. Krom toho jsou zde umístěny čtyři přepínatelné červené body. Poloha zvoleného bodu určuje pozici výstupu, na které závisí výstupní graf přehrávač.

Na každém boxu znázorňující systém je přepínač s polohami „ON“ a „OFF“. Nad schématem je pak přepínač umožňující volbu mezi lineárním a nelineárním systémem.

U obou ze vstupních signálů je možné vybrat jeden ze sedmi předdefinovaných zvuků. Tři z nich jsou generovány oscilátorem: sinus o výšce tónu c^2 (523.2 Hz), sinus o výšce g^2 (784 Hz) a pila o výšce a^1 (440 Hz). Další čtyři jsou nahrané ze souborů. Jedná se o zvuk malého bubnu, nezkreslené a zkreslené elektrické kytary a bílý šum.

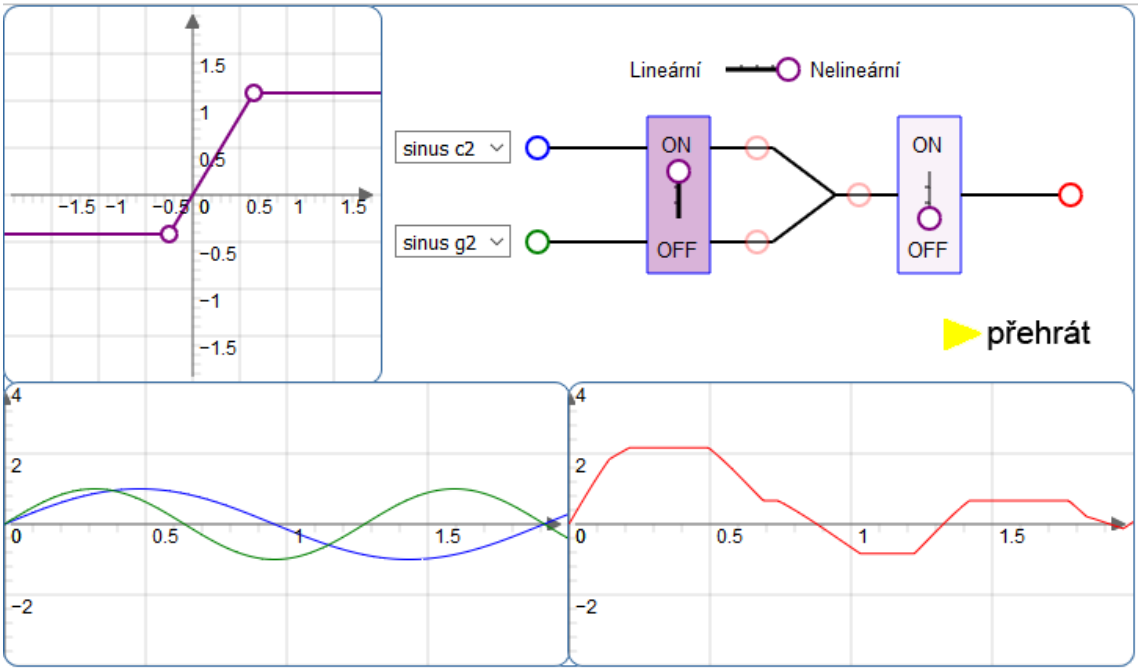
V levém horním rohu je graf převodní funkce. Ten obsahuje dva body, jejichž pohybem je možné převodní funkci nastavit. To se děje tak, aby jejich spojnice vždy procházela počátkem. Pohyb s jedním bodem tedy mění o polohu druhého bodu, ovšem pouze v rámci osy x. Poloha na ose y zůstává konstantní.

Pokud je aktivován lineární systém, funkce má tvar přímky a určuje pouze zesílení signálu. Při přepnutí na nelineární systém poloha bodů na ose y představuje Treshold limiteru, tedy hranici nad (respektive pod) kterou je všechen signál uřezán. Treshold je možné nastavit zvlášť pro kladné a zvlášť pro záporné části signálu.

Ve spodní části apletu jsou dva grafy, první zobrazuj vstupní signály (jeden modře a druhý zeleně), druhý pak výstupní signál. Ten podle polohy vybraného bodu výstupu může představovat buď jeden ze vstupních signálů po průchodu systémem ale

před jejich sečtením, signál po sečtení, nebo signál po sečtení a po průchodu druhým systémem. Oba grafy zobrazují průběh signálu v časové oblasti a mají rozsah 2 ms pro generované zvuky nebo 200 ms pro zvuky nahrané ze souboru. Vzhledem k tomu, že se podle typu zdroje signálu liší jeho měřítko, graf výsledného součtu signálů různých typů by nedával smysl. Proto se při takovém nastavení výstupní graf zobrazí jako dva různé nesečtené grafy, přestože přehrávaný signál vypadá jinak.

Posledním prvkem je tlačítko pro ovládání přehrávače, který vždy přehrává signál v místě vybraného výstupního bodu.



Obrázek 7.2: Rozhraní apletu lineární a nelineární systémy

7.6 Implementace

Pro vykreslování rozhraní apletu je opět použita knihovna JSXGraph. Schéma je složeno z geometrických elementů náležících k hlavnímu rámečku. Výjimku opět tvoří políčka pro výběr signálu, které jsou vytvořeny jako HTML element Input.

Jelikož je zde velké množství ovládacích prvků, které se navíc často musejí chovat rozdílně v závislosti na nastavení ostatních ovládacích prvků, byla vytvořena jedna centrální funkce, kterou volají při změně všechny ovládací prvky a jako parametry

předávají své identifikační číslo a nastavenou hodnotu. Funkce se pak sama postará o zavolání správných funkcí v návaznosti na stav celého apletu.

U převodní funkce představovalo největší problém definovat její chování při přechodu mezi jednotlivými nulovými hodnotami. Přechod přes osu x byl zamezen, přímo na ose x dochází automaticky k nastavení zesílení na 0 a druhý bod je přemístěn do polohy `[0;0]`. To byl jediný způsob, jak zamezit odchýlení spojovací přímky od počátku grafu.

Přechod přes osu y je umožněn. Oba body však vždy musí být v protějších kvadrantech. Přímo na ose y je teoretické zesílení nekonečné. Aby se ovšem zamezilo dělení nulou, je zavolání funkce, která počítá zesílení i funkce pro ovládání pohybu druhého bodu, přeskočeno.

Waveformy do grafu se vykreslují stejným způsobem jako u grafu v apletu Filtrace signálu, zobrazení efektivních hodnot prvních 200 ms, tentokrát za pomocí sto osmdesáti svislých čar.

K přehrávání a zpracování zvukového signálu je opět stejně jako u apletu Filtrace signálu využito rozhraní Web Audio API. Konkrétně jako zdroj zvuků je využit modul `OscilatorNode`, nebo přehrání bufferu nahraného ze souboru typu wav. Pro zpracování signálu je využit `GainNode` pro úpravu hlasitosti a `ScriptProcessorNode`, ve kterém byl implementován limiter. Zdroje zvuků jsou vždy ve dvou instancích, jednou pro každou ze dvou větví vstupních signálů, sekce pro zpracování jsou ve třech instancích (dva pro každou větev samostatně a třetí po jejich sečtení). První dvě jsou ale vždy ovládány společně.

Tyto moduly jsou vždy řazeny za sebe a případně zapojeny do výstupního uzlu `context.destination` v závislosti na nastavení apletu.

Původní plán zobrazovat kromě grafů časové oblasti i grafy spekter byl zamítnut, kvůli vysoké hardwarové náročnosti. Vyžadovalo by to provádět Fourierovu transformaci na třech místech zvlášť, což je na hranici současných možností jazyka javascript.

8 ZÁVĚR

V této práci byly shrnuty teoretické poznatky pro tvorbu šesti apletu v jazyce javascript. Dále byla popsána jejich implementace a uživatelské prostředí.

První kapitola se zabývá obecně použitým programovacím jazykem, jeho vznikem, historií a základními vlastnostmi. Dále pak pro tvorbu apletů využitých knihoven, v prvé řadě opensource knihovnou JSXGraph, sloužící k vykreslování grafů a geometrických objektů.

Druhá kapitola je věnována apletu kombinace signálů. Je zde vysvětlena lineární a konvexní kombinace signálů a popsáno rozhraní a jeho implementaci. Popis implementace se zaměřuje na základní rozvržení architektury apletu a hlavní problémy.

Ve třetí kapitole je podrobně rozebráno téma sčítání harmonických funkcí, zvláště pak harmonických funkcí se stejnou frekvencí. Dále je zde popis rozhraní apletu a popis implementace, který se zaměřuje na vysvětlení nejdůležitějších funkcí knihovny JSXGraph.

Čtvrtá kapitola je zaměřena na filtraci signálu, obsahuje vysvětlení frekvenčních spekter, harmonické analýzy a popis jednotlivých typů filtrů. I zde je popis rozhraní apletu a popis implementace, tentokrát se zaměřením na základy rozhraní Web Audio API.

Obsahem páté kapitoly je vliv fáze na audiosignál. Je zde rozebrán vliv fáze na tvar signálu a na jeho vnímání lidským sluchem. Jedna podkapitola je věnovaná frekvenčnímu spektru obdélníkového signálu. Opět je přiložen popis rozhraní apletu a stručný popis implementace.

Šestá kapitola popisuje jev zvaný aliasing a jeho vliv na zvukový signál při vzorkování. Na konci kapitoly nechybí popis rozhraní apletu a implementace.

Poslední, sedmá kapitola se zabývá systémy, jejich rozdělením na lineární a nelineární a vysvětlením časově proměnných a invariantních systémů. Kapitola je zakončena popisem apletu, který vysvětluje toto téma. U implementace jsou popsány hlavní problémy, které se vyskytly při tvorbě apletu.

Jako výstup tedy bylo vytvořeno šest apletů. Ke každému z nich byla vytvořena jednoduchá webová stránka popisující ovládání a základy teorie. U dvou z těchto apletů (konkrétně u apletu Filtrace signálů a Lineární a nelineární systémy) není běh zcela plynulý. To je dáno omezeními výkonu, která vyplývají z principu jazyka Javascript.

LITERATURA

[1] SUEHRING, Steve. *JavaScript: krok za krokem*. Brno: Computer Press, 2008. Krok za krokem (Computer Press). ISBN 978-80-251-2241-9.

[2] PURCELL, Lee a Mary Jane MARA. *JavaScript: tvorba dokonalých www stránek : podrobný průvodce začínajícího uživatele*. Praha: Grada, 1998. Průvodce (Grada). ISBN 80-7169-531-9.

[3] BAŠE, Ondřej. *JQuery pro neprogramátory: průvodce využitím knihovny jQuery UI*. Brno: Computer Press, 2012. ISBN 978-80-251-3750-5.

[4] *JSXGraph* [online]. [cit. 2016-12-04]. Dostupné z: http://jsxgraph.uni-bayreuth.de/wp/index.html

[5] JS Doc Reference [online]. [cit. 2016-12-04]. Dostupné z: http://kryzalid91.free.fr/js/converter/docs/index.html

[6] VRBA, Kamil. *Analogová technika.* Elektronické skriptum. Brno: FEKT VUT v Brně, 2011.

[7] SCHIMMEL, Jiří. *Studiová a hudební elektronika*. Elektronické skriptum. Brno: FEKT VUT v Brně, 2015. ISBN 978-80-214-4452-2

[8] Lineární kombinace. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2016-12-10]. Dostupné z: https://cs.wikipedia.org/wiki/Line%C3%A1rn%C3%AD_kombinace

[9] BARTSCH, Hans-Jochen. *Matematické vzorce*. Přeložil Zdeněk TICHÝ. Praha: Státní nakladatelství technické literatury, 1983.

[10] SMÉKAL, Zdeněk. *Analýza signálů a soustav – BASS*. Elektronické skriptum. Brno: FEKT VUT v Brně 2012. ISBN 978-80-214-4453-9.

[11] SEDLÁČEK, Jiří. *Elektrotechnika II*. Elektronické skriptum. Brno: FEKT VUT v Brně.

[12] Harmonic Adition Theorem. *Wolfram MathWorld* [online]. [cit. 2017-05-30]. Dostupné z: http://mathworld.wolfram.com/HarmonicAdditionTheorem.html

[13] SCHIMMEL, Jiří. *Elektroakustika*. Elektronické skriptum. Brno: FEKT VUT v Brně, 2016. ISBN 978-80-214-4716-5

[14] MIŠUREC, Jiří. *Číslicové zpracování signálů*. Elektronické skriptum. Brno: FEKT VUT v Brně, 2011.

[15] *Can I Use: Support tables for HTML5, CSS3, etc* [online]. [cit. 2017-05-20]. Dostupné z: http://caniuse.com/#feat=audio-api

[16] JAN, Jiří. *Číslicové zpracování a analýza signálů: stručné skriptum*. Brno: Vysoké učení technické v Brně, 2010. ISBN 978-80-214-4018-0.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

ECMA	European Computer Manufacturers Association, Evropská asociace výrobců počítačů
DOM	Document Object Model, Objektový model dokumentu
HTML	HyperText Markup Language, Hypertextový značkovací jazyk
CSS	Cascading Style Sheets, Šablony kaskádových stylů
DP	Dolní propust
HP	Horní propust
PP	Pásmová propust
PZ	Pásmová zádrž
FR	Fourierův rozvoj, Fourierovy řady
FT	Fourierova transformace
FFT	Fast Fourier Transform, Rychlá Fourierova transformace
$u(x)$	Funkce jednotkového skoku
DAW	Digital Audio Workstation
A-D	Analog – Digitál
D-A	Digitál – Analog
LGPL	Lesser General Public License
LTI	Linear Time Invariant, lineární časově invariantní

A OBSAH ELEKTRONICKÉ PŘÍLOHY

Na přiloženém CD se nacházejí html soubory obsahující zdrojové kódy jednotlivých appletů, pomocné soubory, zvukové samply a javascriptové knihovny nutné pro jejich spuštění a elektronická verze této práce. Obsaženy jsou tyto soubory:

- BakalářskáPráce.pdf – elektronická verze této práce
- applet1.1.html – aplet Součet sinus a cosinus
- applet2.html – aplet Kombinace signálů
- applet3.html – aplet Aliasing a jeho projevy
- applet4.5.html – aplet Filtrace signálu
- applet5.html – aplet Význam fáze u audiosignálu
- applet6.1.html – aplet Lineární a nelineární systémy
- dsp.js – javascriptová knihovna pro FFT
- gibberish.js – javascriptová knihovna pro jednoduché přehrání zvuku
- Guitar1.wav – sample čisté elektrické kytary
- Guitar2.wav – sample zkreslené elektrické kytary
- index.html – webová stránka s rozcestníkem apletů
- jsxgraph.css – soubor kaskádových stylů nezbytný pro knihovnu jsxGraph
- jsxgraphcore.js – javascriptová knihovna pro vykreslování geometrických objektů a grafů
- playButton.png – obrázek tlačítka
- Snare.wav – sample úderu do bubnu
- stopButton.png – obrázek tlačítka
- style.css – soubor kaskádových stylů pro stránky apletů
- whitenoise.wav – sample bílého šumu

Vzhledem k tomu, že kód javascriptu je nedílnou součástí html souborů, nejsou zdrojové kódy znovu přiloženy samostatně.

Dočasně jsou také applety umístěny na adrese http://www.bilbophoto.cz/applets/, později budou přidány do společného seznamu apletů Ústavu telekomunikací na http://www.utko.feec.vutbr.cz/~rajmic/applets/.